

AD-A171 524

DEVELOPMENT OF AN ATMOSPHERIC DISPERSION MODEL FOR  
HEAVIER-THAN-AIR GAS M. (U) ARKANSAS UNIV FAYETTEVILLE  
DEPT OF CHEMICAL ENGINEERING J A HAVENS ET AL MAY 85

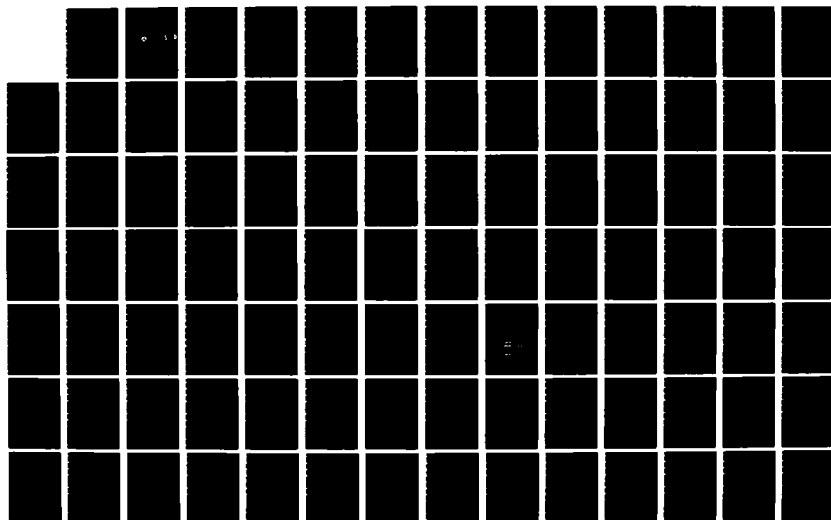
1/3

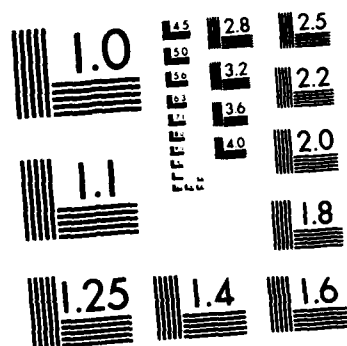
UNCLASSIFIED

USCG-D-24-85 DTCG23-88-C-28829

F/G 28/4

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

Report No. CG-D-24-85

**DEVELOPMENT OF AN  
ATMOSPHERIC DISPERSION MODEL  
FOR HEAVIER-THAN-AIR GAS MIXTURES**

**Volume III:  
DEGADIS User's Manual**



JERRY A. HAVENS

THOMAS O. SPICER

DTIC  
ELECTE  
SEP 02 1986  
S D

This report is available to the U.S. public through the National  
Technical Information Service, Springfield, Virginia 22161.

**FINAL REPORT  
MAY 1985**

Prepared for:

**U.S. Department of Transportation  
United States Coast Guard**

**Office of Research and Development  
Washington, D.C. 20593**

**DISTRIBUTION STATEMENT A**

**Approved for public release  
Distribution Unlimited**

86 9 2 040

AD-A171 524

DTIC FILE COPY

### NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

The contents of this report do not necessarily reflect the official view or policy of the Coast Guard; and they do not constitute a standard, specification, or regulation.

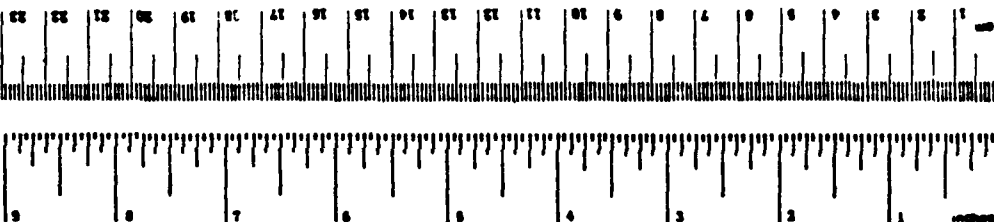
This report, or portions thereof may not be used for advertising or sales promotion purposes. Citation of trade names and manufacturers does not constitute endorsement or approval of such products.



1. Report No. CG-D-24-85	2. Government Accession No. <b>AD-A171524</b>	3. Recipient's Catalog No.	
4. Title and Subtitle DEVELOPMENT OF AN ATMOSPHERIC DISPERSION MODEL FOR HEAVIER-THAN-AIR GAS MIXTURES Volume III: DEGADIS User's Manual		5. Report Date May 1985	
		6. Performing Organization Code	
7. Author(s) Jerry A. Havens and Thomas O. Spicer		8. Performing Organization Report No. Final Report	
9. Performing Organization Name and Address Chemical Engineering Department University of Arkansas Fayetteville AR 72701		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. DT-CG-23-80-C-20029	
12. Sponsoring Agency Name and Address U.S. Coast Guard Commandant (G-FCP-22F/TP64) 2100 Second Street, SW Washington DC 20593		13. Type of Report and Period Covered Final Report Sept. 1980-May 1985	
		14. Sponsoring Agency Code	
15. Supplementary Notes Final Report is in <u>three</u> volumes: Volumes I, Development of an Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures, and Volume II, Laboratory Calm Air Heavy Gas Dispersion Experiments, are bound separately.			
16. Abstract  The mathematical modeling techniques used to predict atmospheric dispersion of heavier-than-air gases discussed in Volume <u>I</u> of <del>this report</del> are briefly summarized; these techniques are incorporated in an interactive computer model DEGADIS. Details of <del>the</del> DEGADIS implementation are briefly discussed.  The necessary input information to simulate a heavier-than-air gas release with DEGADIS is summarized. Example simulations of a steady state and transient release are included. A list of DEGADIS self-diagnostics with suggested actions are included.  A listing of DEGADIS is included along with a partial list of program variables. Guidelines for installation of DEGADIS are presented.			
17. Key Words Heavy Gas Dispersion Turbulent Mixing Atmospheric Dispersion Buoyancy-Driven Flows Stratified Flows		18. Distribution Statement Gravity-Spreading Flows Density Currents Dispersion Models Risk Assessment	
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages 278	22. Price

# METRIC CONVERSION FACTORS

Approximate Conversions from Metric Measures			
When You Know	Multiply by	To Find	Symbol
<b>LENGTH</b>			
millimeters	0.00	inches	in
centimeters	0.4	inches	cm
meters	3.3	feet	m
kilometers	1.1	miles	km
kilometers	0.6	miles	km
<b>AREA</b>			
square centimeters	0.16	square inches	cm <sup>2</sup>
square meters	1.2	square yards	m <sup>2</sup>
square kilometers	0.4	square miles	km <sup>2</sup>
hectares (10,000 m <sup>2</sup> )	2.5	acres	ha
<b>MASS (weight)</b>			
grams	0.05	ounces	g
kilograms	2.2	pounds	kg
tonnes (1000 kg)	1.1	short tons	t
<b>VOLUME</b>			
milliliters	0.03	fluid ounces	ml
liters	2.1	pints	l
liters	1.06	quarts	l
liters	0.26	gallons	l
cubic meters	35	cubic feet	m <sup>3</sup>
cubic meters	1.3	cubic yards	m <sup>3</sup>
<b>TEMPERATURE (heat)</b>			
Celsius temperature	5/9 (then add 32)	Fahrenheit temperature	°C
Fahrenheit temperature	5/9 (then subtract 32)	Celsius temperature	°F



Approximate Conversions to Metric Measures			
When You Know	Multiply by	To Find	Symbol
<b>LENGTH</b>			
inches	2.5	centimeters	in
feet	30	centimeters	ft
yards	0.9	meters	y
miles	1.6	kilometers	mi
<b>AREA</b>			
square inches	6.5	square centimeters	in <sup>2</sup>
square feet	0.09	square meters	ft <sup>2</sup>
square yards	0.8	square meters	y <sup>2</sup>
square miles	2.6	square kilometers	mi <sup>2</sup>
acres	0.4	hectares	a
<b>MASS (weight)</b>			
ounces	28	grams	oz
pounds	0.45	kilograms	lb
short tons (2000 lb)	0.9	tonnes	st
<b>VOLUME</b>			
teaspoons	5	milliliters	sp
tablespoons	15	milliliters	tblsp
fluid ounces	30	milliliters	fl oz
cups	0.24	liters	c
pints	0.47	liters	p
quarts	0.95	liters	q
gallons	3.8	liters	gal
cubic feet	0.03	cubic meters	ft <sup>3</sup>
cubic yards	0.76	cubic meters	y <sup>3</sup>
<b>TEMPERATURE (heat)</b>			
Fahrenheit temperature	5/9 (then subtract 32)	Celsius temperature	°F
Celsius temperature	5/9 (then add 32)	Fahrenheit temperature	°C

1 in = 2.54 cm (exact). For other exact conversions and more detailed tables, see NIST Spec. Publ. 286, Units of Length and Measure, Price \$2.25, MD Catalog No. C13.10 286.



Dist and/or Special

A-1

## TABLE OF CONTENTS

	<u>Page</u>
List of Tables	vii
List of Figures	ix
List of Symbols	xi
Summary	1
I.    DEGADIS Model Summary	3
1.    Summary Description	3
2.    Model Limitations and Cautions	6
II.   DEGADIS Model Inputs	11
1.    VAX/VMS Command Procedure	11
2.    Simulation Definition	13
III.  DEGADIS Model Implementation and Outputs	19
1.    Input Module--DEGADISIN	19
2.    Source Module--DEGADIS1	21
3.    Pseudo-Steady State Module--DEGADIS2	28
4.    Observer Time-Sort Module--DEGADIS3	32
5.    Steady State Module--SDEGADIS2	38
IV.   Example Simulations	43
1.    Example Input Session	44
2.    Simulation Output	53
References	57
Appendix A:  DEGADIS Model Installation on VAX/VMS	A-1
B:    Considerations for Installation Other than VAX/VMS	B-1
C:    Code Listing	C-1
D:    Error Messages	D-1
E:    Partial Listing of Program Variables	E-1

## LIST OF TABLES

	<u>Page</u>
I.1. Typical Values of Surface Roughness	7
I.2. Representative Monin-Obukhov Lengths and Power Law Exponents for Different Atmospheric Stabilities	8
IV.1. Summary of BURRO9 Test Conditions Used in Example Simulation	43

## LIST OF FIGURES

	<u>Page</u>
I.1. Schematic Diagram of DEGADIS Model	4
II.1. Example DEGADIS Command Procedure on VAX/VMS for a Steady State Simulation Named BURRO9S	12
II.2. Example DEGADIS Command Procedure on VAX/VMS for a Transient Simulation Named BURRO9	12
II.3. Summary of Simulation Definition Input Information for DEGADIS	15
III.1. DEGADISIN Flow Chart	20
III.2. Structure of Free-Formatted RUNNAME.INP File	21
III.3. DEGADIS1 Flow Chart	22
III.4. SYS\$DEGADIS:EXAMPLE.ER1 Listing	26
III.5. DEGADIS2 Flow Chart	29
III.6. SYS\$DEGADIS:EXAMPLE.ER2 Listing	33
III.7. DEGADIS3 Flow Chart	34
III.8. SYS\$DEGADIS:EXAMPLE.ER3 Listing	37
III.9. SDEGADIS2 Flow Chart	39
IV.1. BURRO9S.INP Listing	54
IV.2. BURRO9.INP Listing	55
IV.3. DEGADIS-Predicted Centerline Maximum Concentration vs. Maximum Reported Concentration--BURRO9	56

# LIST OF SYMBOLS

$b$	half width of horizontally homogeneous central section of gas plume (m)
$C_{p_c}$	heat capacity of contaminant (J/kg K)
$c_c$	centerline, ground level concentration ( $\text{kg/m}^3$ )
$E$	plume strength (kg/s)
$E(\tau)$	source rate (kg/s)
$H$	height or depth of density intrusion or cloud (m)
$H_a$	ambient absolute humidity (kg water/kg bone dry air)
$h$	enthalpy of source blanket (J/kg)
$h_0$	overall heat transfer coefficient ( $\text{J/m}^2 \text{ s K}$ )
$k$	von Karman's constant, 0.35
$L$	source length (m)
$M_i$	initial cloud mass (kg)
$p$	atmospheric pressure (atm)
$q_1$	empirical constant in heat capacity equation
$R$	gas source cloud radius (m)
$R_m$	value of $R$ when $(\pi R^2 Q_*)$ is a maximum (m)
$R_{\text{max}}$	maximum radius of the cloud (m)
$R_p$	primary source radius (m)
$S_{z0}$	$S_z$ at the downwind edge of the source ( $x = L/2$ ) (m)
$S_{z0_m}$	value of $S_{z0}$ when $(\pi R^2 Q_*)$ is a maximum (m)

$T$	temperature associated with source blanket enthalpy (K)
$T_0$	contaminant storage temperature (K)
$T_s$	surface temperature (K)
$t$	time (s)
$u_a$	ambient average velocity (m/s)
$u_e$	horizontal or frontal entrainment velocity (m/s)
$u_*$	friction velocity (m/s)
$V_h$	heat transfer velocity (0.0125 m/s) (m/s)
$w_a$	mass fraction of air
$w_c$	mass fraction of contaminant
$w_e$	vertical entrainment velocity associated with $H_L$ (m/s)
$x, y, z$	Cartesian coordinates (m)
$z_0$	reference height in wind velocity profile specification (m)
$z_R$	surface roughness (m)
$\alpha$	constant in power law wind profile
$\beta$	constant in $\sigma_y$ correlation
$\Gamma$	gamma function
$\gamma$	ratio of $(\rho - \rho_a)/c_c$
$\delta$	constant in $\sigma_y$ correlation
$\lambda$	Monin-Obukhov length (m)
$\rho$	density of gas/air mixture (kg/m <sup>3</sup> )
$\rho_0$	density of contaminant's saturated vapor at $T_0$ (kg/m <sup>3</sup> )
$\psi$	logarithmic velocity profile correction function

## SUMMARY

The mathematical modeling techniques used to predict atmospheric dispersion of heavier-than-air gases discussed in Volume I of this report are briefly summarized; these techniques are incorporated in an interactive computer model DEGADIS.

The necessary input information to simulate a heavier-than-air gas release is summarized. Example simulations of a steady state and transient release are included. Guidelines for installation of DEGADIS are included, and a listing of DEGADIS is included along with a partial list of program variables.



## I. DEGADIS MODEL SUMMARY

The DEGADIS model methods, equations, and supporting data are described in detail in Volume I of this report. This section is intended to summarize the critical components of the model formulation and the associated limitations, and to indicate cautions and diagnostic guidelines which should be followed in its use. The suggested limitations and guidelines are based on the experience gained during the development of the model and its verification by comparison with a wide range of heavier-than-air gas dispersion tests. These limitations and precautions will almost certainly be refined through further application of the model.

### I.1. Summary Description

The DEGADIS model combines the principal features of the Shell HEGADAS model (Colenbrander, 1980, and Colenbrander and Puttock, 1983) and a box model proposed by van Ulden (1983). DEGADIS incorporates some features not contained in either of the original models and substitutes methods which we believe are more appropriate for treatment of other features. The general application of the model involves formation of a "secondary" gas source, the subsequent entrainment of gas from that secondary source by the wind field, and downwind dispersion of the gas plume or cloud. Figure I.1 illustrates the general methodology; the description of the formation and development of the secondary source utilizes the box model, and the entrainment from the secondary source and subsequent downwind dispersion utilizes the similarity representations of the cloud concentration and vertical velocity profiles of the HEGADAS model. Heavier-than-air gas releases which cannot be represented as steady, continuous releases are modeled as a series of pseudo-steady releases.

Application of the model to releases of a heavier-than-air gas in zero wind involves only the box model. The box model

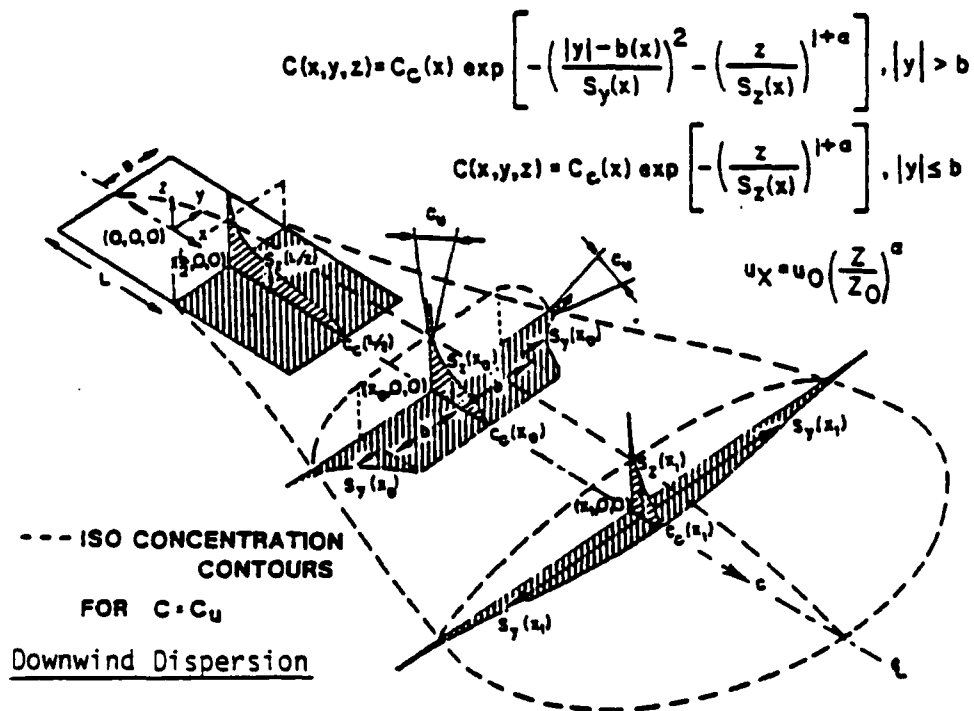
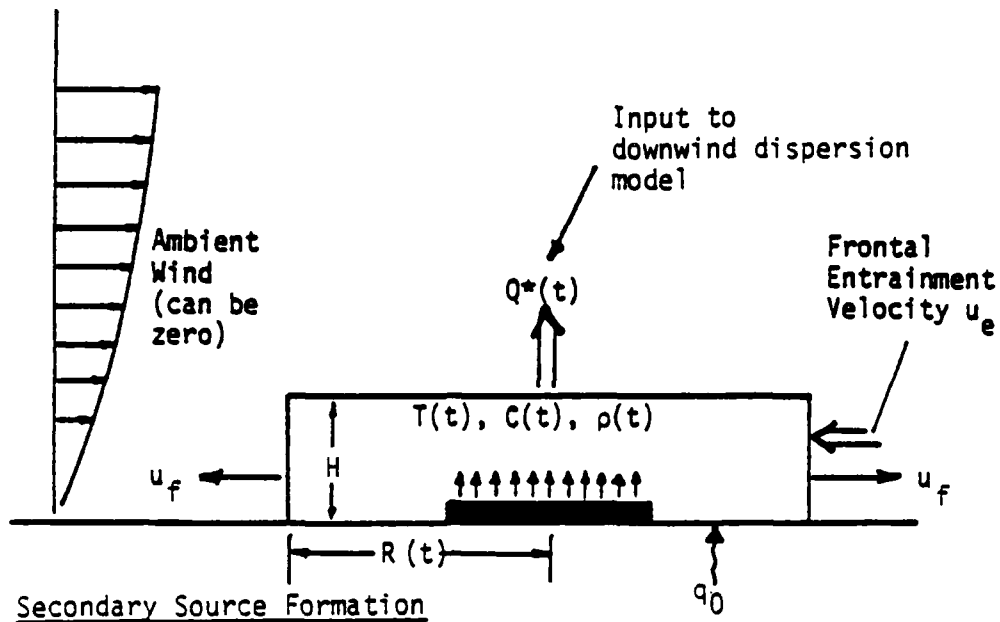


Figure I.1. Schematic diagram of DEGADIS model.

treatment of gravity spreading and associated air entrainment is based on parameterization of the laboratory still-air experiments described in Volume II of this report. For releases in wind, the box model also describes the source development but provides for entrainment of the gas-air source cloud by the action of the wind. The rate of release relative to the wind velocity determines the important characteristics of the cloud dispersion process. For high rates of release (instantaneous being the most rapid) in low wind, the buoyancy-dominated flow processes described by the box model predominate and may essentially determine downwind distances for dilution of the gas cloud to the 1-10% gas concentration levels which characterize the lower flammability limit for light hydrocarbons. Conversely, for low rates of release in high winds, the rate of entrainment due to gravity spreading is not very important, and the downwind dispersion process is controlled primarily by the vertical dispersion produced by the action of the wind field shear in the cloud. The treatment of the large range of "intermediate" conditions (i.e. where gravity spreading and air entrainment by the secondary source importantly influence the source cloud concentration and dimensions, and consequently the "initial condition" for downwind dispersion calculation) distinguishes the DEGADIS model.

DEGADIS incorporates heat transfer and water transfer when applicable from the underlying surface to the cloud. Inclusion of these procedures in the model is optional. Effects of heat transfer on both the mean cloud buoyancy and the vertical turbulent mixing (air entrainment) are included while direct effect of water transfer is included only in the mean cloud buoyancy.

DEGADIS is written in Digital Equipment Corporation's VAX/VMS\* Fortran (a superset of ANSI Fortran 77); it is

---

\*VAX and VMS are registered trademarks of Digital Equipment Corporation.

composed of five programs which communicate using ASCII files (see Section II). A listing of the code is included as Appendix C, and a partial list of program variables is given in Appendix E. Considerations for installation of DEGADIS are discussed in Appendices A and B. DEGADIS self-diagnostics are listed in Appendix D along with suggested actions.

### I.2. Model Limitations and Cautions

DEGADIS model application should be limited to the description of atmospheric dispersion of heavier-than-air gas releases at ground level onto flat, unobstructed terrain or water. Application to releases from sources above ground level (e.g. overflow from dikes) would be expected to give conservative predictions of the downwind hazard zones, but this has not been verified.

The dispersion of a heavier-than-air gas by the action of the wind assumes the maintenance of a wind velocity profile in the gas cloud or plume whose characteristics are determined by the approach wind flow (upwind of the release). The treatment of vertical momentum transfer invokes the assumption of a logarithmic vertical velocity profile, which is in turn curve-fitted to a power law vertical velocity profile. DEGADIS also uses similarity forms for the vertical profile of gas concentration in the cloud, and the vertical profile is dependent on the power law exponent  $\alpha$  used in the representation of the velocity profile. The vertical velocity profile, which is directly related to the air entrainment velocity into the cloud, is dependent on the factors which determine the structure of the atmospheric boundary surface layer, wind speed, surface roughness, and atmospheric stability. Consequently, the representations of the vertical velocity and concentration profiles in DEGADIS are subject to similar limitations as in other descriptions of the surface layer. Table I.1 indicates typical recommended surface roughness values. Table I.2 indicates logarithmic wind

velocity profile corrections for different atmospheric stabilities, along with typical values of the wind profile power law exponent  $\alpha$  determined in DEGADIS.

TABLE I.1  
TYPICAL VALUES OF SURFACE ROUGHNESS

Terrain	$z_{R,M}$
Mud flats, ice	$10^{-5}$
Calm, open sea	$10^{-4}$
Off-sea wind in coastal areas	$10^{-3}$
Cut grass (~ 3 cm)	0.007
Long grass (~ 60 cm), crops	0.04

Demonstration of the model has been primarily directed to the prediction of hazard extent defined by gas concentrations in the hydrocarbon flammable limit range (-1 to 20%). Even though the relation between peak gas concentration and time-averaged gas concentration is uncertain, there is some basis for using 2.0 as an estimate of the peak-to-time-averaged-concentration ratio for determining a flammable gas concentration zone. If this assumption is made, the predicted distance to LFL/2 would be the maximum distance at which a flammable gas concentration would be predicted. Based on the simulations of field experiments presented in Volume I, the ratio of observed distance to calculated distance for a given time-averaged concentration level (OBS/PRE) ranged from 0.82 to 1.03 for the 2.5% level nine out of ten times (i.e. 90%).

TABLE I.2  
REPRESENTATIVE MONIN-OBUKHOV LENGTHS AND POWER LAW EXPONENTS  
FOR DIFFERENT ATMOSPHERIC STABILITIES

Pasquill Stability Category	Monin-Obukhov Length ( $\lambda$ ) as a Function of Surface Roughness $z_R$ (m)	Typical (*) Power Law Exponents $\alpha$	Corrections to Logarithmic Profiles as Given by Businger (1973)
A	-11.4 $z_R^{0.10}$	0.108	$\psi = 2 \ln \left[ \frac{1+a}{2} \right] + \ln \left[ \frac{1+a^2}{2} \right]$
B	-26.0 $z_R^{0.17}$	0.112	$- 2 \tan^{-1}(a) + \pi/2$
C	-123 $z_R^{0.30}$	0.120	with $a = (1 - 15(z/\lambda))^{1/4}$
D	$\infty$	0.142	$\psi = 0$
E	123 $z_R^{0.30}$	0.203	$\psi = -4.7(z/\lambda)$
F	26.0 $z_R^{0.17}$	0.253	

(\*) calculated for 8 m/s at 10 m with surface roughness of 0.001 m

confidence interval); for the 5% level, (OBS/PRE) ranged from 0.73 to 0.96 for a 90% confidence interval. If for a given release scenario the calculated distance to the 2.5% average concentration level was 120 m, the distance to the 2.5% average concentration for nine out of ten realizations of the same release would be expected to range between 98 m and 124 m, which would also represent the range of the downwind extent of the flammable gas concentration zone for LNG if the peak-to-average ratio of 2.0 is assumed.

## II. DEGADIS Model Inputs

As implemented under VAX/VMS, DEGADIS uses three areas of input information:

- (\*) VAX/VMS command procedure for execution
- (\*) simulation definition
- (\*) numerical parameters

The VAX/VMS command procedure used to execute DEGADIS can be generated in DEGADISIN. As well, DEGADISIN is the interactive input module which provides the simulation definition. An example input session is included in Section IV.2. The numerical parameters (convergence criteria, initial increments, etc.) are supplied to DEGADIS through a series of input files. Although these numerical parameters are easily changed, the user should need to change these only rarely with the exception of the time sort parameters which are explained in Section II.3.

### II.1. VAX/VMS Command Procedure

The VAX/VMS command procedure generated by DEGADISIN controls the execution of images for the simulation. Image execution follows one of two paths, either for a transient release or for a steady state release. DEGADISIN will automatically generate the appropriate command procedure; but first, DEGADISIN requests a simulation name be specified. The simulation name must be a valid VAX/VMS file name without a file extension and is designated RUNNAME. DEGADIS will use this file name with standard extensions for input, interprocess communication, and output. Figures II.1 and II.2 show example VAX/VMS command procedures for the run name BURRO9S for steady state and BURRO9 transient releases, respectively. The directory which contains the executable images of DEGADIS has been assigned the system logical name SYS\$DEGADIS (see Appendix A). The COPY/LOG command simply copies a file from the first argument to the second argument, and the RUN command executes



the specified image. Of course, these steps may also be carried out by issuing the commands at a terminal.

```
$COPY/LOG  SYS$DEGADIS:EXAMPLE.ER1  BURRO9S.ER1
$COPY/LOG  SYS$DEGADIS:EXAMPLE.ER2  BURRO9S.ER2
$RUN  SYS$DEGADIS:DEGADIS1
BURRO9S
$RUN  SYS$DEGADIS:SDEGADIS2
BURRO9S
$COPY/LOG  BURRO9S.SCL+BURRO9S.SR3-
BURRO9S.LIS
```

Figure II.1. Example DEGADIS command procedure on VAX/VMS for a steady state simulation named BURRO9S.

```
$COPY/LOG  SYS$DEGADIS:EXAMPLE.ER1  BURRO9.ER1
$COPY/LOG  SYS$DEGADIS:EXAMPLE.ER2  BURRO9.ER2
$COPY/LOG  SYS$DEGADIS:EXAMPLE.ER3  BURRO9.ER3
$RUN  SYS$DEGADIS:DEGADIS1
BURRO9
$RUN  SYS$DEGADIS:DEGADIS2
BURRO9
$RUN  SYS$DEGADIS:DEGADIS3
BURRO9
$COPY/LOG  BURRO9.SCL+BURRO9.SR3-
BURRO9.LIS
```

Figure II.2. Example DEGADIS command procedure on VAX/VMS for a transient simulation named BURRO9.

## II.2. Simulation Definition

DEGADISIN is an interactive method of simulation definition where the user specifies information about the ambient wind field, the properties of the released gas, and some details of the release.

The ambient wind field is characterized by a known velocity  $u_0$  at a given height  $z_0$ , a surface roughness  $z_R$ , and the Pasquill stability class. The Pasquill stability class is used to estimate values of the lateral similarity parameter coefficients  $\delta$  and  $\beta$  (Pasquill, 1974), values of the along-wind similarity coefficients (Beals, 1971), and the Monin-Obukhov length  $\lambda$  used by Businger et al. (1971) in their logarithmic velocity profile function  $\Psi$  (Section I). The Monin-Obukhov length is then used to calculate the friction velocity  $u_*$ . Once these parameters have been estimated using the Pasquill stability class, the user has the option of interactively changing any of these to better describe the simulation. In addition to these specifications, the ambient temperature, pressure, and humidity must be specified.

The properties of air and the released gas are used to evaluate the mixture density as a function of temperature and composition. The desired released gas properties include the molecular weight  $MW_c$ , the storage temperature (normal boiling point for cryogenic gases)  $T_0$ , the vapor phase density at the storage temperature and ambient pressure  $\rho_0$ , and two constants  $q_1$  and  $p_1$  which describe the heat capacity according to the equation

$$C_{p_c}(T) = MW_c \left[ 3.33 \times 10^4 + q_1 \left[ \frac{p_1}{T - T_0} \right] \right] \quad (\text{II-1})$$

where  $C_{p_c}(T)$  is the mean heat capacity (J/kg K) at temperature

T. Note that a constant heat capacity with respect to

temperature can be obtained by setting  $p_1 = 1.0$  and choosing the appropriate value for  $q_1$ . Representative gas properties for liquefied natural gas (LNG) as methane and liquefied petroleum gas (LPG) as propane are included in DEGADISIN. Also included are the lower and upper flammability limits (LFL and UFL, respectively) for LNG and LPG.

The user may also choose to calculate the mixture density as a function of composition using some other method. This mixture density is entered in the program as if the release were isothermal; for each composition, the program requests the contaminant mole fraction, the contaminant concentration, and the mixture density. For ease of input, these values may be entered from a file made available to DEGADISIN.

In specifying the details of the release, the user must choose to simulate the release as transient or steady state. For both release types, the area source is assumed circular. The source radius and emission rate must be specified for a steady state release only once, while these must be specified as a function of time for transient releases (either interactively or by file). For transient releases, the user must specify the initial amount of gas present over the source in order to simulate an instantaneous release (e.g. the Thorney Island Trials).

Figure II.3 summarizes the simulation information gathered by DEGADISIN contained in the RUNNAME file with extension INP. The structure of RUNNAME.INP is illustrated in Figure III.2. At this point, RUNNAME.INP may be edited to correct any misinformation entered during the input session. Note that care must be exercised when editing RUNNAME.INP due to the fact that information contained in the file can be different depending on the answered questions (e.g. steady state versus transient simulation).

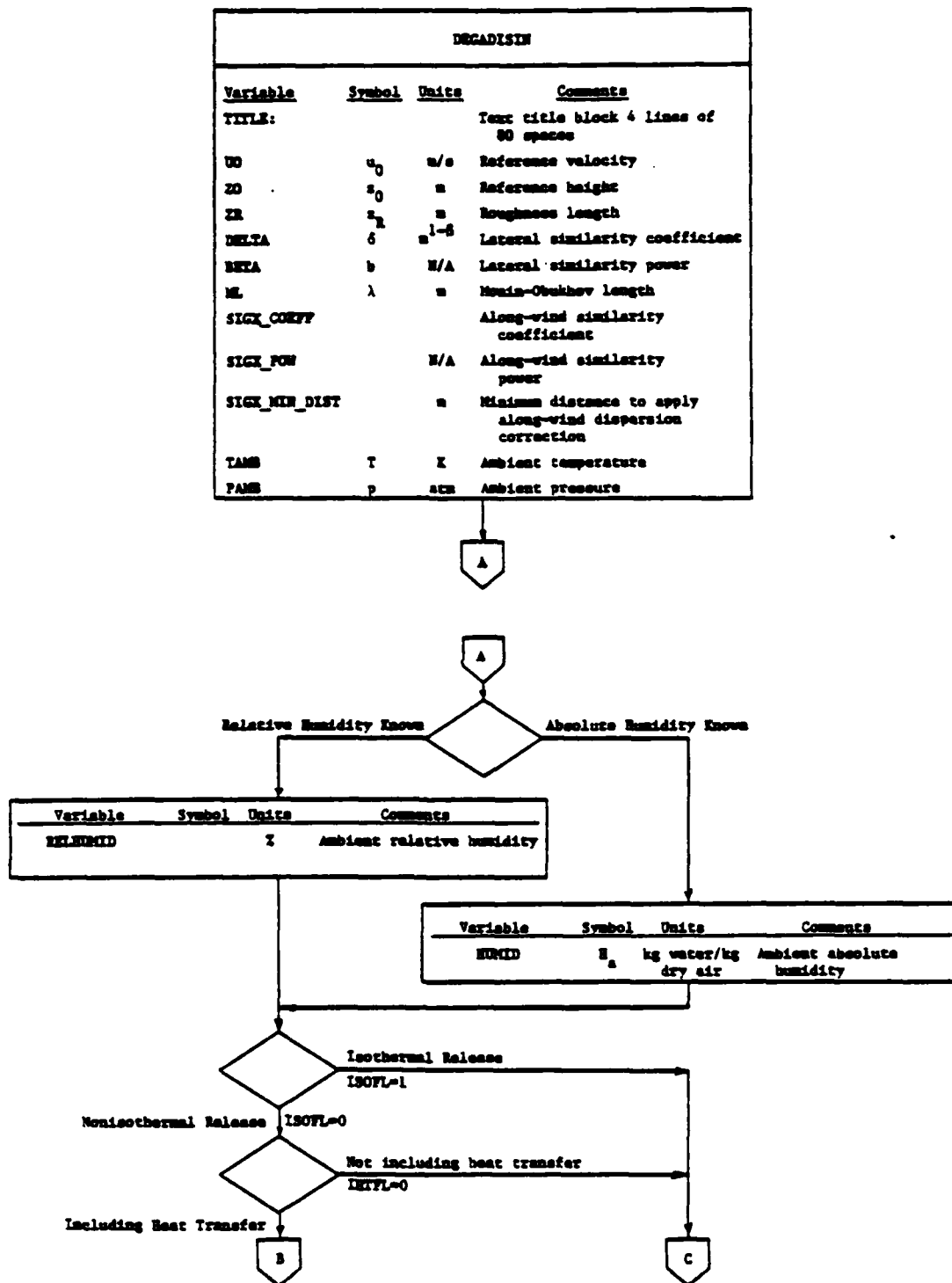


Figure II.3. Summary of simulation definition input information for DEGADIS.

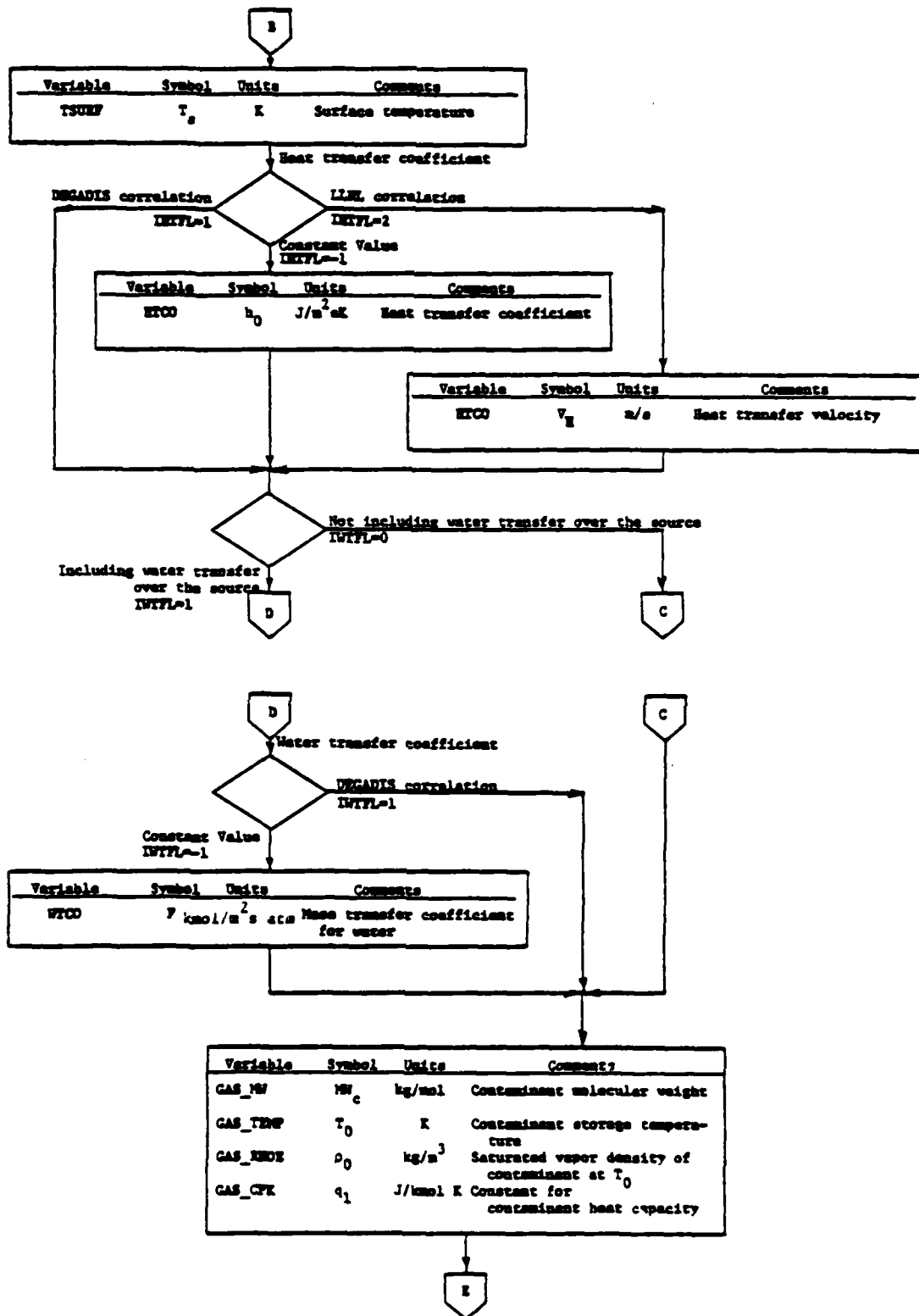


Figure II.3. (continued)

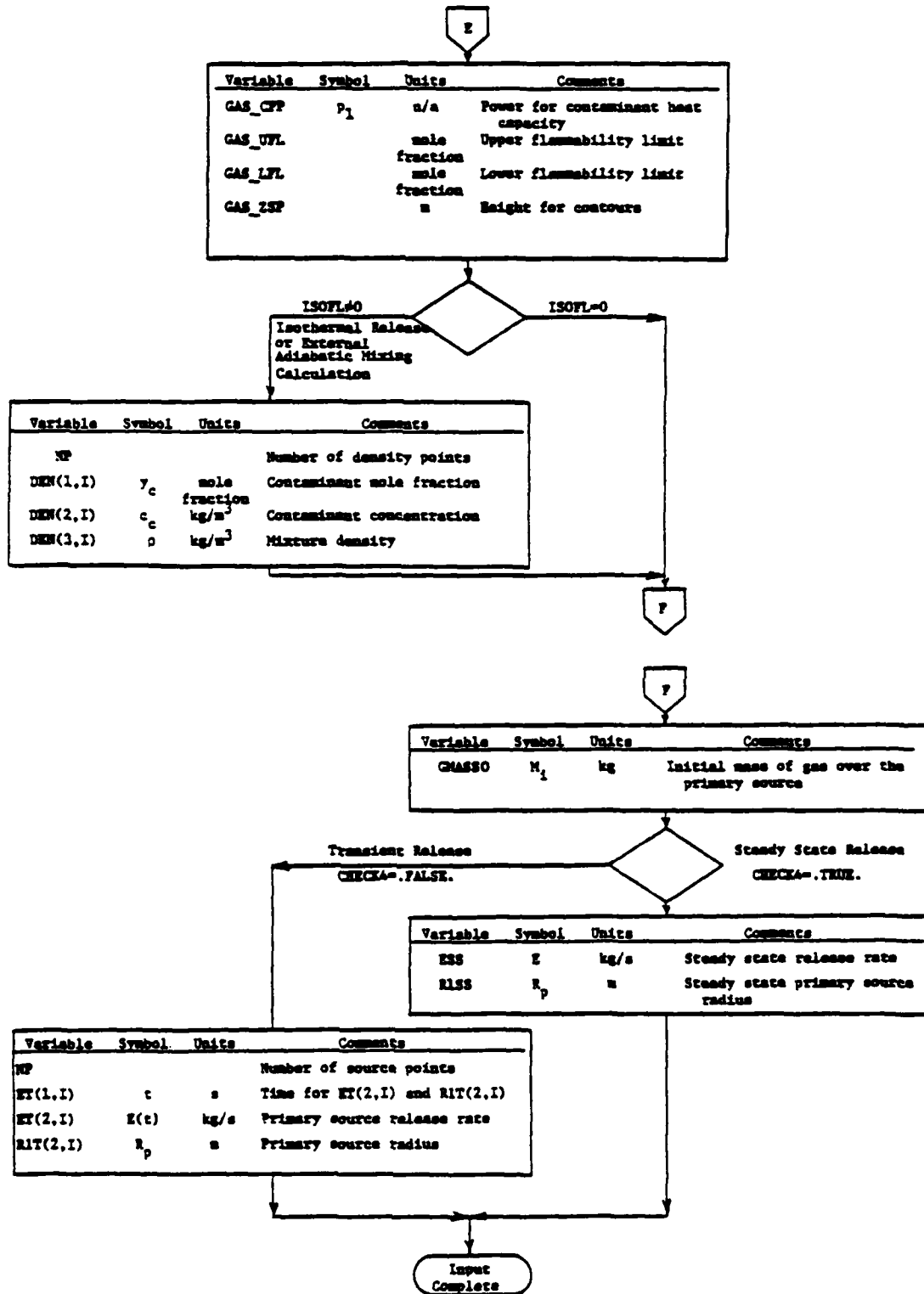


Figure II.3. (continued)



### III. Model Implementation and Outputs

The model described in Section I has been implemented in VAX/VMS Fortran (a superset of Fortran 77) in the code DEGADIS. DEGADIS is comprised of five separate programs as follows:

- (\*) DEGADISIN is the interactive input module which defines the simulation.
- (\*) DEGADIS1 determines  $\alpha$  and describes the gas source for transient and steady state releases.
- (\*) DEGADIS2 describes the pseudo-steady state downwind dispersion of the released gas.
- (\*) DEGADIS3 sorts the results of DEGADIS2 for a transient release.
- (\*) SDEGADIS2 describes the steady state downwind dispersion of the released gas.

As indicated in Figures II.1 and II.2, a steady state release is simulated by executing DEGADISIN, DEGADIS1, and SDEGADIS2, while a transient release is simulated by executing DEGADISIN, DEGADIS1, DEGADIS2, and DEGADIS3.

#### III.1 Input Module--DEGADISIN

DEGADISIN (see Section II) is the interactive input module which defines the simulation; DEGADISIN is composed of two subroutines (Figure III.1):

- (\*) DEGADISIN contains the program overhead and generates the command file RUNNAME.COM which can be used to control simulation execution (C-29).



(\*) IOT contains the interactive question and answer sequence which defines the simulation; IOT also creates the file RUNNAME.INP (C-55).

An example of a DEGADISIN query sequence is included in Section IV.2. As this information is gathered, it is written to the file RUNNAME.INP (see Figure III.2). Once DEGADISIN is completed, RUNNAME.INP may be edited to correct minor input mistakes. If major revisions are necessary, the recommended practice is to execute DEGADISIN again.

Once the information required by DEGADISIN has been entered properly, DEGADIS may be executed using the command procedure generated by DEGADISIN under the file name RUNNAME.COM. If DEGADIS is not to be run using this command file, the user must enter the simulation name (RUNNAME) after each of the programs are begun. As well, the user must provide copies of the numerical parameter files.

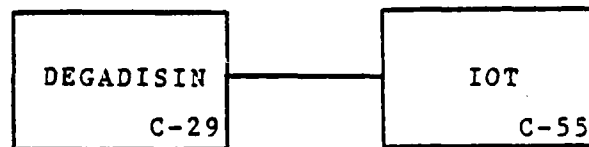


Figure III.1. DEGADISIN flow chart.

```

TITLE(1)
TITLE(2)
TITLE(3)
TITLE(4)
UO, ZO, ZR
ISTAB
DELTA, BETA, ML
SIGX_COEFF, SIGX_POW, SIGX_MIN_DIST
TAMB, PAMB, HUMID
ISOFL, TSURF
IHTFL, HTCO
IWTFL, WTCO
GAS_NAME
GAS_MW, GAS_TEMP, GAS_RHOE
GAS_CPK, GAS_CPP
GAS_UFL, GAS_LFL, GAS_ZSP
NP
If (ISOFL=0) then (for external density calculations) NP of these
{
DEN((J,1),J=1,5)
DEN((J,2),J=1,5)
.
.
.
DEN(J,NP),J=1,5)
CCLOW
GMASSO
NT
{
ET(1,1), ET(2,1), R1T(2,1)
ET(1,2), ET(2,2), R1T(2,2)
.
.
.
ET(1,NT), ET(2,NT), R1T(2,NT)
CHECK1, CHECK2, AGAIN, CHECK3, CHECK4,
CHECK5
TINP
for steady state only ESS, SLEN, SWID

```

Figure III.2. Structure for free-formatted RUNNAME.INP file.

### III.2 Source Module--DEGADIS1

DEGADIS1 estimates values for the friction velocity and ambient wind profile power  $\alpha$  and characterizes the primary gas source for the remainder of the model; DEGADIS1 is composed of the following subroutines (Figure III.3):

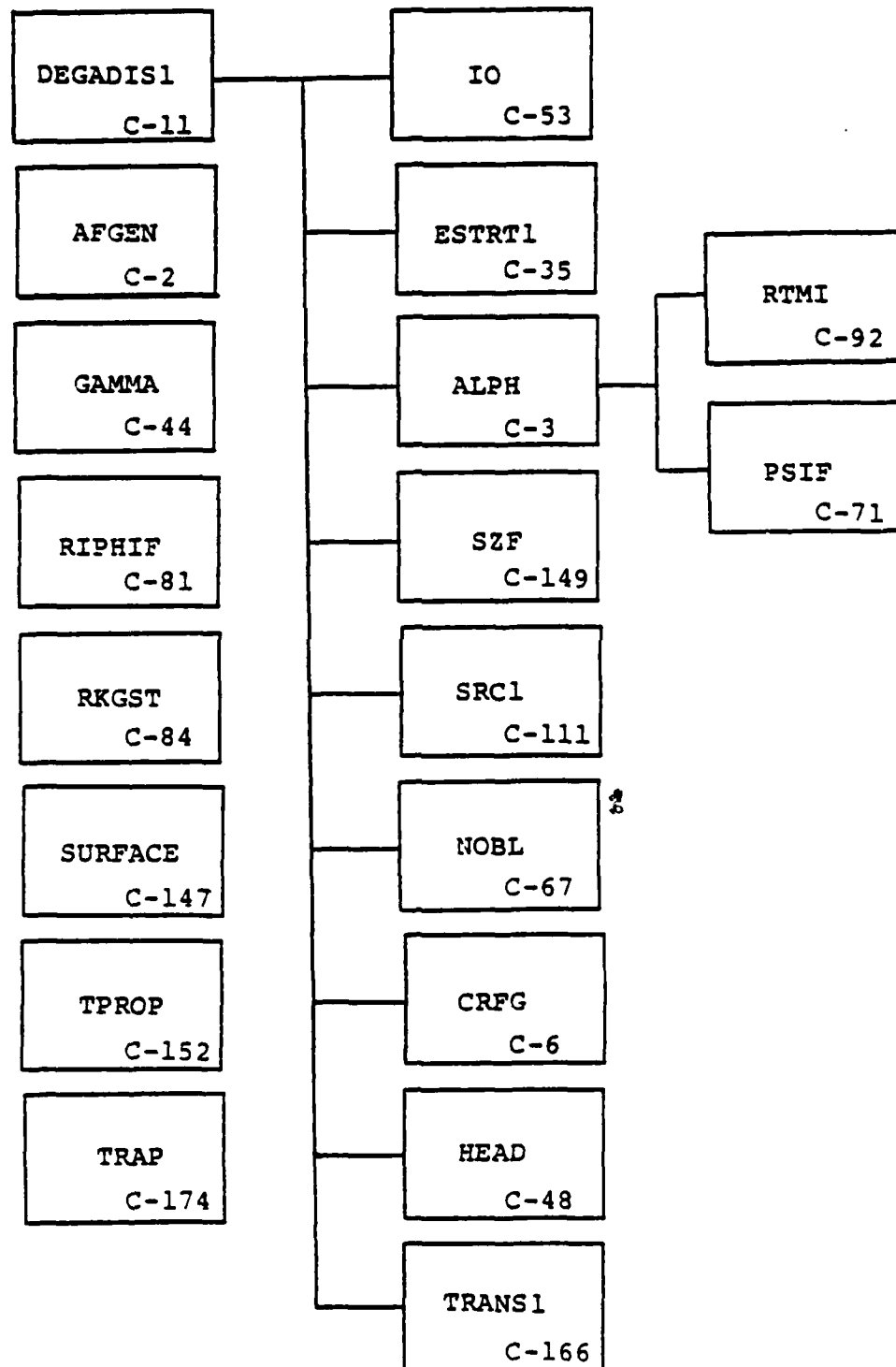


Figure III.3. DEGADIS1 flow chart.

- (\*) AFGEN is a utility which linearly interpolates between a pair of points based on a list of supplied values (C-2).
- (\*) ALPH estimates the ambient wind profile power  $\alpha$  by minimizing the integral of the difference between an ambient logarithmic velocity profile and the assumed power law velocity profile (C-3).
- (\*) CRFG creates a table of calculated values which will describe the secondary gas source for the downwind dispersion calculations (C-6).
- (\*) DEGADIS1 contains the program overhead and sequentially calls the routines required to estimate the ambient wind profile power  $\alpha$  and to characterize the primary gas source (C-11).
- (\*) ESTR1 recovers the numerical parameters contained in the file RUNNAME.ER1 (C-35).
- (\*) GAMMA is a utility function that calculates the gamma function of the argument (i.e.  $\Gamma(x)$ ) (C-44).
- (\*) HEAD writes a formatted output heading to the file RUNNAME.SCL (C-48).
- (\*) IO recovers the simulation definition contained in RUNNAME.INP (C-53).

- (\*) NOBL            estimates gas source behavior when no gas blanket is present (C-67).
  
- (\*) PSIF            calculates the  $\rho$  function in the logarithmic velocity profile (C-71).
  
- (\*) RIPHIF          is a series of utilities which calculates the Richardson number and the value of  $\phi(Ri)$  (C-81).
  
- (\*) RKGST          is a utility routine which performs numerical integration of a specified system of equations using a modified fourth order Runge-Kutta method (C-84).
  
- (\*) RTMI            is a utility routine which solves the trial and error set up by ALPH (C-92).
  
- (\*) SRC1            contains the ordinary differential equations which describe the gas blanket formed as a result of the primary gas source (C-111).
  
- (\*) SURFACE        is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (C-147).
  
- (\*) SZF            estimates the value of  $S_z$  if the primary source can just form a gas blanket over the source (C-149).
  
- (\*) TPROP          is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (C-152).

- (\*) TRANS1      writes the information to continue the next simulation step to the file RUNNAME.TR2 (C-166).
- (\*) TRAP        is a utility included for program diagnostics (C-174).

As input, DEGADIS1 requires two files:

- (\*) RUNNAME.ER1 contains various numerical parameters.  
For most simulations, a copy of the SYSSDEGADIS:EXAMPLE.ER1 file will be adequate. See Figure II.1 or II.2. A copy of SYSSDEGADIS:EXAMPLE.ER1 is included in Figure III.4.

As output, DEGADIS1 generates the following files:

- (\*) RUNNAME.SCD contains the calculated values which describe the secondary gas source. It is generated by SRC1 and NOBL and is then read by CRFG; it is a temporary file.
- (\*) RUNNAME.SCL is the listed output which describes the input information for the simulation and the calculated secondary gas source. It is written by HEAD and CRFG.
- (\*) RUNNAME.TR2 contains the information to continue the next simulation step.

! This is an example of how to set up and use the run parameter  
! input files. Comment lines start with an exclamation mark(!)  
! in the first column. The only restrictions for data input are  
! as follows:

- 1) The data must be entered in the same order  
all of the time.
- 2) Only the number must be between columns 10 and 20.
- 3) Always include the decimal point in the number

! Column layout:

! 23456789012345678901234567890

! -----1-----2-----3

	I	I	
STPIN	0.01		MAIN - RKGST - INITIAL STEP SIZE
ERBND	0.0025		MAIN - RKGST - ERROR BOUND
STPMX	5.12		MAIN - RKGST - MAXIMUM STEP SIZE
WTRG	1.		MAIN - RKGST - WEIGHT FOR RG
WTTM	1.		MAIN - RKGST - WEIGHT FOR Total Mass
WTYA	1.		MAIN - RKGST - WEIGHT FOR ya
WTYC	1.		MAIN - RKGST - WEIGHT FOR yc
WTEB	1.		MAIN - RKGST - WEIGHT FOR Energy Balance
WTMB	1.		MAIN - RKGST - WEIGHT FOR Momentum Balance
WTuh	1.		MAIN - RKGST - WEIGHT FOR Ueff*Heff
XLI	0.05		ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
XRI	0.40		ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
EPS	0.001		ALPH - ERROR BOUND USED BY 'RTMI'
ZLOW	0.01		ALPHI - maximum BOTTOM HEIGHT FOR FIT OF ALPHA

! STPINZ -0.02 ALPHI - INITIAL RKGST STEP <0.

! ERBNDZ 0.005 ALPHI - ERROR BOUND FOR RKGST

! STPMXZ -0.04 ALPHI - MAXIMUM STEP FOR RKGST <0.

! Note that comment lines can be mixed with the numbers.

SRCOER	0.007	SRC10 - OUTPUT Error criteria
SRCSS	5.2	SRC10 - min time for Steady; STPMX
SRCcut	.0001	SRC10 - min height for blanket
htcut	.10	SRC1 - min height for blanket heat transfer
ERNOBL	1.0005	NOBL - CONVERGENCE CRITERIA ratio
NOBLPT	100.	NOBL - NUMBER OF POINTS
		USED ON THE LAST PORTION OF THE SOURCE

Figure III.4. SYS\$DEGADIS:EXAMPLE.ER1 listing.

```

crfser      0.008      error criteria in building GEN3 vectors
!
epsilon     0.59       epsilon USED IN AIR ENTRAINMENT SPECIFICATION
!
! /SPRD_CON/
!
ce          1.15       constant in gravity slumping equation
delrhomin   0.025      stop cloud spread if delrho<delrhomin
!
! /SZFC/
!
szsta0      0.01       SZF - Initial step size
szerr       0.001      SZF - Error criteria
szstmax     5.0        SZF - Maximum step size [=] m
szsz0       0.01       SZF - Initial Value of dellay*Ueff*Heff
!
! /ALPHcom/
!
ialpfl      1.         ALPHI - calculation flag; 0) alpha=alpco; 1)1/(1+z); 2)1
alpco       0.2        ALPHI - Value for alpha if IALPFL = 0
!
! /PHIcom/
!
iphifl      3.         PHIF - calc flag
dellay      2.15       Ratio of H1/Heff
!
! /VUcom/
!
vua         1.3        Constant Av in source model
vub         1.2        Constant Bv in source model
vuc         20.0       Constant Ev in source model
vud         .64        Constant Dv in source model
vudelta     0.20       Constant DELTA v in source model
!
! End-of-File

```

Figure III.4. (continued)



### III.3 Pseudo-Steady State Module--DEGADIS2

DEGADIS2 performs the downwind dispersion portion of the calculation for each of several observers released successively over the transient source described by DEGADIS1. DEGADIS2 is composed of the following subroutines (Figure III.5):

- (\*) AFGEN is a utility which linearly interpolates between a pair of points based on a list of supplied values (C-2).
- (\*) DEGADIS2 contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameter file RUNNAME.ER2, and perform the simulation (C-19).
- (\*) ESTR22 recovers the numerical parameters contained in the file RUNNAME.ER2, particularly the number of observers NOBS (C-39).
- (\*) OB contains the ordinary differential equations which average the gas source for each observer (C-69).
- (\*) PSS contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when  $b > 0$  (C-72).
- (\*) PSSOUT governs the output of calculated points to the file RUNNAME.PSD when PSS is active (C-75).

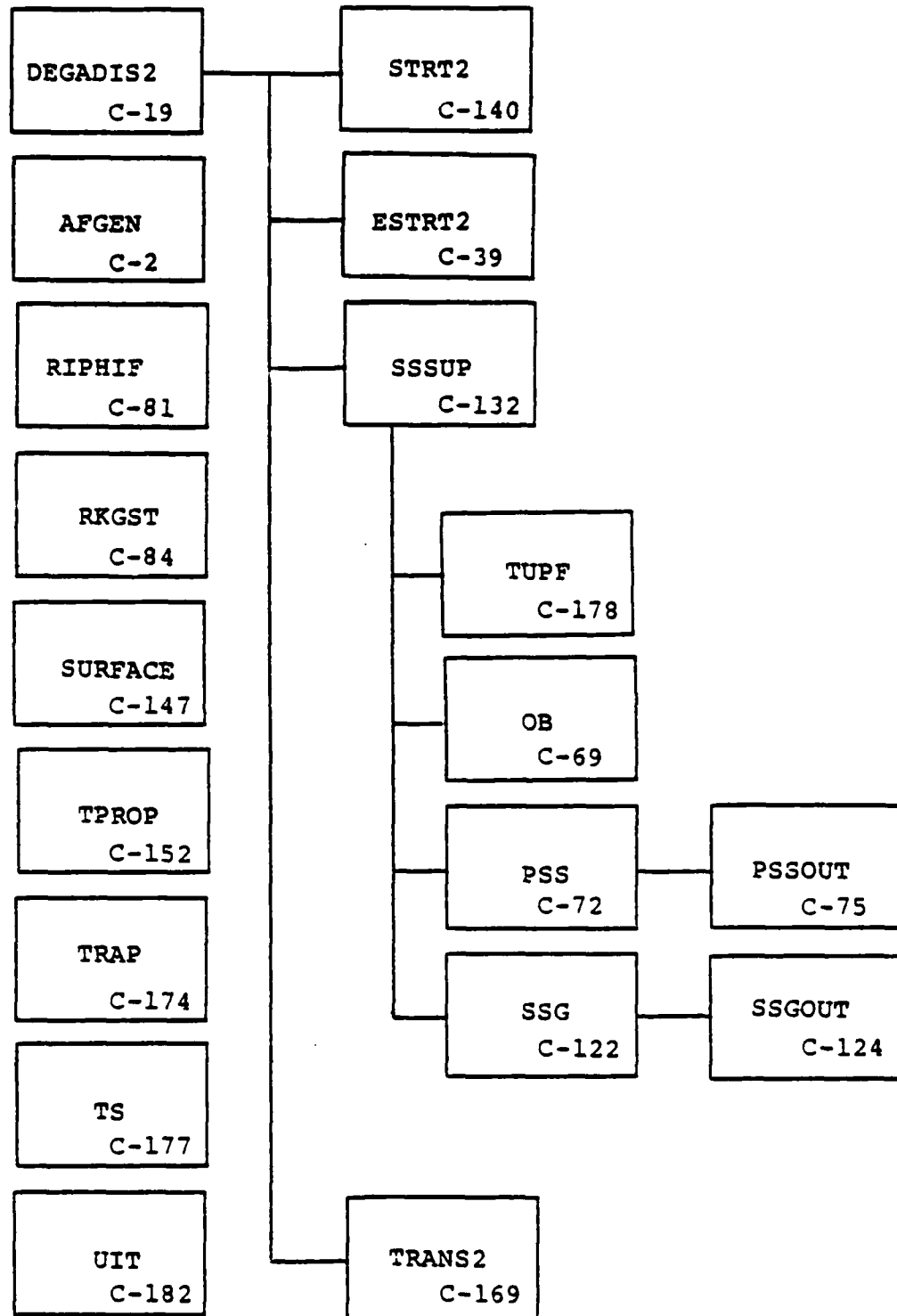


Figure III.5. DEGADIS2 flow chart.

- (\*) RIPHIF is a series of utilities which calculates the Richardson number and the value of  $\phi(Ri)$  (C-81).
- (\*) RKGST is a utility routine which performs numerical integration of a specified system of equations using a modified fourth order Runge-Kutta method (C-84).
- (\*) SSG contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when  $b = 0$  (C-122).
- (\*) SSGOUT governs the output of calculated points to the file RUNNAME.PSD when SSG is active (C-124).
- (\*) SSSUP is a supervisor routine which controls the averaging of the source for each observer, the portion of the downwind dispersion calculation when  $b > 0$ , and the portion of the downwind dispersion calculation when  $b = 0$  (C-132).
- (\*) STRT2 recovers the information generated in DEGADIS1 contained in the file RUNNAME.TR2 (C-140).
- (\*) SURFACE is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (C-147).

- (\*) TPROP is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (C-152).
- (\*) TRANS2 writes the information necessary for DEGADIS3 to the file (RUNNAME.TR3) (C-169).
- (\*) TRAP is a utility included for program diagnostics (C-174).
- (\*) TS calculates the time when a given observer will be at a given downwind distance (C-177).
- (\*) TUFF contains the two routines which determine the intersection of the upwind/downwind edge of the secondary gas source with a given observer (C-178).
- (\*) UIT is a series of routines to calculate observer position and velocity as a function of time (C-182).

As input, DEGADIS2 requires two files:

- (\*) RUNNAME.ER2 contains various numerical parameters, particularly the number of observers NOBS. For most simulations, a copy of the SYS\$DEGADIS:EXAMPLE.ER2 file will be adequate. See Figure II.1 or II.2. A copy of SYS\$DEGADIS:EXAMPLE.ER2 is included in Figure III.6

(\*) RUNNAME.TR2 contains the basic simulation definition as well as calculated secondary source parameters.

DEGADIS2 generates the following output files:

(\*) RUNNAME.PSD contains the calculated downwind dispersion parameters for each observer. DEGADIS3 sorts this information to determine the downwind concentration profiles as a function of time.

(\*) RUNNAME.TR3 contains the simulation definition and the number of each record type written to RUNNAME.PSD.

#### III.4. Time Sort Module--DEGADIS3

DEGADIS3 sorts the downwind dispersion calculation for each of several observers for concentration information at several given times; the along-wind dispersion correction is then applied as desired. DEGADIS3 uses the following subroutines (Figure III.7):

(\*) DEGADIS3 contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS2, recover the numerical parameter file RUNNAME.ER3, sort and apply the along-wind dispersion correction to the results of DEGADIS2, and output the results (C-24).

! This is an example for an 'ER2' run parameter file.  
! The same rules apply as for the 'ER1' files.

! 23456789012345678901234567890

! -----1-----2-----3

! These values are in common area /ERROR/

* SYOER	0.03	SSSUP - RKGST - INITIAL SY
ERRO	0.005	SSSUP - RKGST(OBS) - ERROR BOUND
SZOER	0.01	SSSUP - RKGST(OBS) - INITIAL SZ
WTAIO	1.0	SSSUP - RKGST(OBS) - WEIGHT FOR AI
WTQOO	1.0	SSSUP - RKGST(OBS) - WEIGHT FOR Q
WTSZO	1.0	SSSUP - RKGST(OBS) - WEIGHT FOR SZ
* ERRP	0.003	SSSUP - RKGST(PSS) - ERROR BOUND
* SMXP	10.	SSSUP - RKGST(PSS) - MAXIMUM STEP
* WTSZP	1.0	SSSUP - RKGST(PSS) - WEIGHT FOR SZ
* WTSYP	1.0	SSSUP - RKGST(PSS) - WEIGHT FOR SY
* WTBEP	1.0	SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
* WTDH	1.0	SSSUP - RKGST(PSS) - WEIGHT FOR DH
* ERRG	0.003	SSSUP - RKGST(SSG) - ERROR BOUND
* SMXG	10.	SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
ERTDNF	0.0005	TDNF - CONVERGENCE CRITERIA
ERTUPF	0.0005	TUPF - CONVERGENCE CRITERIA
* WTRUH	1.0	SSSUP - RKGST(SSG) - WEIGHT FOR RUH
* WTDHG	1.0	SSSUP - RKGST(SSG) - WEIGHT FOR DH

! These values are in common area /STP/

STPO	0.05	SSSUP - RKGST(OBS) - INITIAL STEP
* STPP	0.05	SSSUP - RKGST(PSS) - INITIAL STEP
* ODLF	0.06	SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
* ODLLP	80.	SSSUP-RKGST(PSS)-MAXIMUM DISTANCE BETWEEN OUTPUTS(m)
* STPG	0.05	SSSUP - RKGST(SSG) - INITIAL STEP
* ODLG	0.045	SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
* ODLLG	80.	SSSUP-RKGST(SSG)-MAXIMUM DISTANCE BETWEEN OUTPUTS(m)

! The last variable NOBS is in /CNOBS/

! Note: it is read in as a real value even though it is integer type  
! in the program.

NOBS 30.

! End-of-File

\*used by steady state simulation

Figure III.6. SYSSDEGADIS:EXAMPLE.ER2 listing

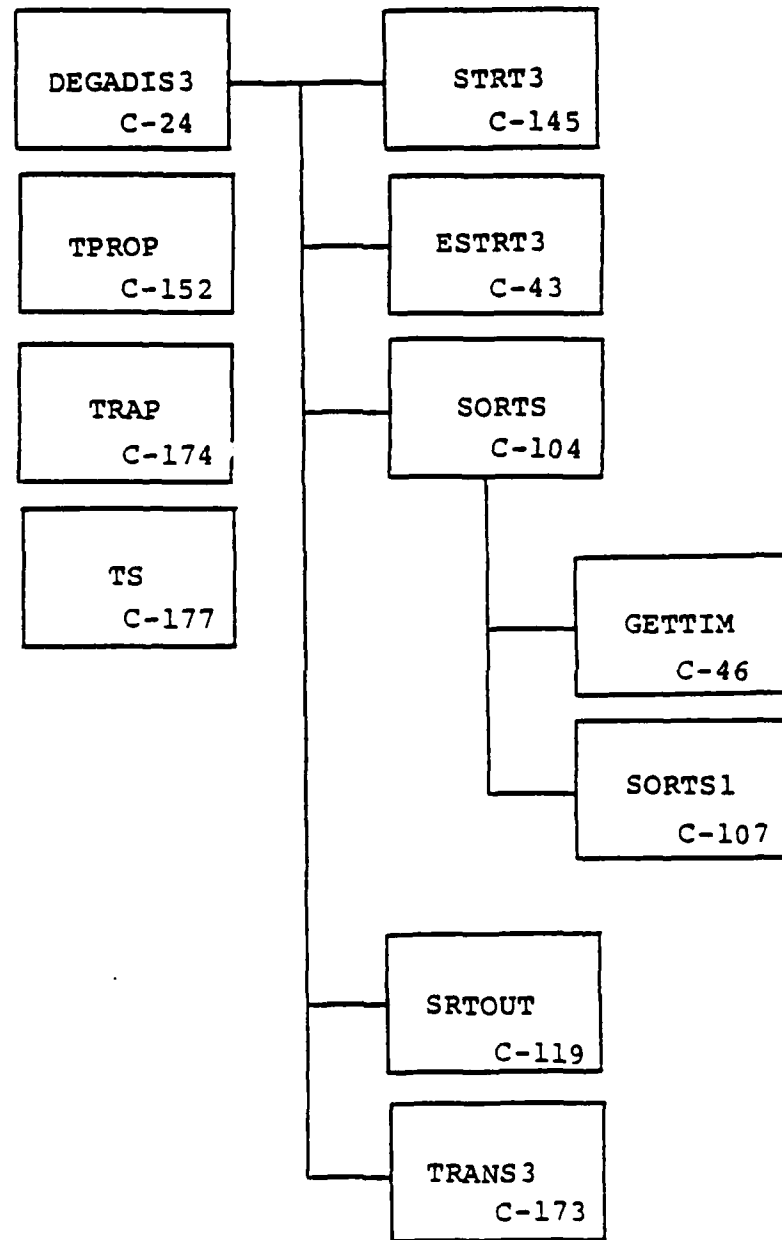


Figure III.7. DEGADIS3 flow chart.

- (\*) ESTRT3      recovers the numerical parameters contained in the file RUNNAME.ER3, particularly the time sort parameters (C-43).
- (\*) GETTIM      sets the default time sort parameters as needed (C-46).
- (\*) SORTS      recovers the information in RUNNAME.PSD and arranges the information according to the time sort parameters in the file RUNNAME.ER3 (C-104).
- (\*) SORTS1      applies the along-wind dispersion correction to the time-sorted information (C-107).
- (\*) SRTOUT      generates the formatted output file RUNNAME.SR3 (C-119).
- (\*) STRT3      recovers the information generated in DEGADIS2 contained in the file RUNNAME.TR3 (C-145).
- (\*) TPROP      is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (C-152).
- (\*) TRANS3      writes RUNNAME.TR4 which contains the necessary information to recover the other output files for this simulation (C-173).
- (\*) TRAP      is a utility included for program diagnostics (C-174).



- (\*) TS calculates the time when a given observer will be at a given downwind distance (C-177).

As input, DEGADIS3 requires three files:

- (\*) RUNNAME.ER3 contains various numerical parameters including the time sort parameters and the flag which dictates whether the along-wind dispersion correction is applied. A copy of SYS\$DEGADIS:EXAMPLE.ER3 file uses the default time sort parameters and includes the along-wind dispersion correction which should apply for most simulations. See Figure II.2. A copy of SYS\$DEGADIS:EXAMPLE.ER3 is included in Figure III.8.
- (\*) RUNNAME.PSD contains the calculated downwind dispersion parameters for each observer. DEGADIS3 sorts this information to determine the downwind concentration profiles as a function of time.
- (\*) RUNNAME.TR3 contains the number of each record type written to RUNNAME.PSD as well as the simulation definition.

! This is an example for an 'ER3' run parameter file.  
! The same rules apply as for the 'ER1' files.

!23456789012345678901234567890

!-----1-----2-----3

! These values are in common area /ERROR/

ERT1	20.	FIRST SORT TIME
ERDT	5.	SORT TIME DELTA
ERNTIM	20.	NUMBER OF TIMES FOR THE SORT

! Note: ERNTIM is entered as a real variable even though  
! it is an integer type variable in the program.

! The value of CHECK5 determines whether the above sort parameters  
! are used. CHECK5 is initialized through the passed transfer  
! files to .FALSE. CHECK5 is set to .TRUE. if a real value of 1.  
! is passed in this file.

CHECK5	0.	USE THE DEFAULT TIME PARAMETERS
!CHECK5	1.	USE THE TIME PARAMETERS GIVEN ABOVE

sixx_flag	1.	correction for x-direction dispersion is to be made
!sixx_flag	0.	no correction for x-direction dispersion

! End-of-File

Figure III.8. SYSSDEGADIS:EXAMPLE.ER3 listing.

As output, DEGADIS3 generates two new files:

- (\*) RUNNAME.SR3 is the formatted output list of the time-sorted concentration parameters. Concentration contours are generated for the specified upper and lower flammability at the specified height entered in DEGADISIN. An example is included in Section IV.
- (\*) RUNNAME.TR4 contains the necessary information to recover the other output files to facilitate further processing.

### III.5 Steady-State Module--SDEGADIS2

SDEGADIS2 is a simplification of DEGADIS2 which uses many of the same subroutines. SDEGADIS2 performs the downwind dispersion portion of the calculation for a steady state source described by DEGADIS1. SDEGADIS2 is composed of the following subroutines (Figure III.9):

- (\*) AFGEN is a utility which linearly interpolates between a pair of points based on a list of supplied values (C-2).
- (\*) ESTR2SS recovers a subset of the numerical parameters contained in the file RUNNAME.ER2 (C-41).
- (\*) PSS is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when  $b > 0$  (C-72).

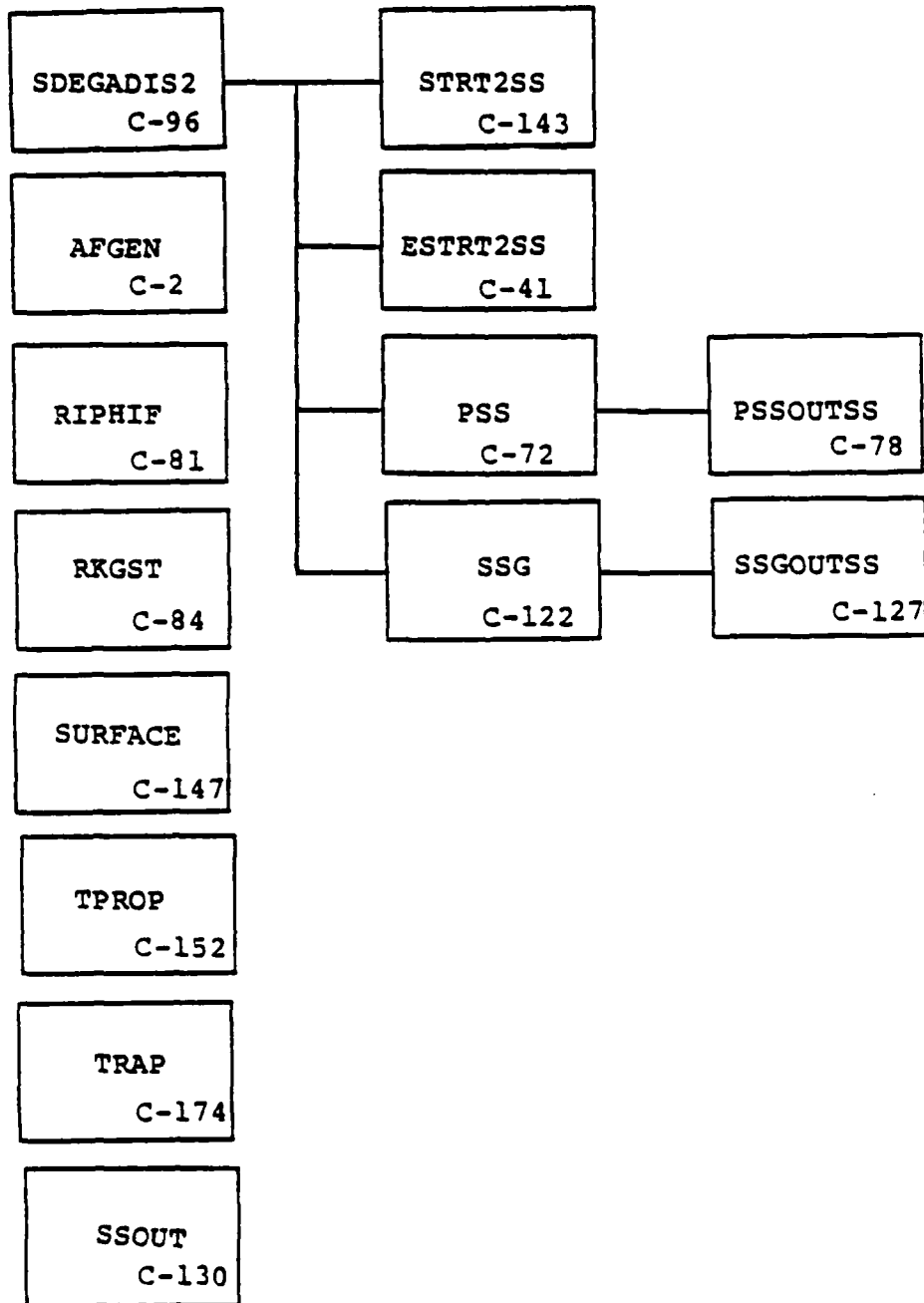


Figure III.9. SDEGADIS2 flow chart.

- (\*) PSSOUTSS governs the output of calculated points to the file RUNNAME.SR3 when PSS is active (C-78).
- (\*) RIPHIF is a series of utilities which calculates the Richardson number and the value of  $\phi(Ri)$  (C-81).
- (\*) RKGST is a utility routine which performs numerical integration of a specified system of equations using a modified fourth order Runge-Kutta method (C-84).
- (\*) SDEGADIS2 contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameters file RUNNAME.ER2, and perform the steady state simulation (C-96).
- (\*) SSG is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when  $b = 0$  (C-122).
- (\*) SSGOUT governs the output of calculated points to the file RUNNAME.SR3 when SSG is active (C-124).
- (\*) SSOUT writes RUNNAME.SR3 and calculates the concentration contours (C-130).

- (\*) **STRT2SS**      recovers a subset of the information generated in DEGADIS1 contained in the file **RUNNAME.TR2** (C-143).
- (\*) **SURFACE**      is a utility routine which estimates heat and water transfer rates across the bottom surface of the gas layer (C-147).
- (\*) **TPROP**        is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (C-152).
- (\*) **TRAP**         is a utility included for program diagnostics (C-174).

As input, SDEGADIS2 require two files:

- (\*) **RUNNAME.ER2** contains various numerical parameters; the steady state simulation requires only part of these. For most simulations, a copy of the **SYS\$DEGADIS:EXAMPLE.ER2** file will be adequate. See Figure II.1. A copy of **SYS\$DEGADIS:EXAMPL.ER2** is included in Figure III.6.
- (\*) **RUNNAME.TR2** contains the basic simulation definition as well as calculated secondary source parameters; the steady state simulation requires only part of these.

As output, SDEGADIS2 generates the following files:

- (\*) RUNNAME.SR3 is the formatted output list of the downwind concentration parameters. Concentration contours are generated for the specified upper and lower flammability at the specified height entered in DEGADISIN.
- (\*) RUNNAME.TR3 contains the necessary information to recover the other output files to facilitate further processing.

#### IV. EXAMPLE SIMULATIONS

In 1980, the U.S. Department of Energy sponsored at China Lake, California, a series of nine LNG releases referred to as the BURRO series of experiments (Koopman et al., 1982). The input conditions (Table IV.1) for the numerical examples in this sections are those of BURRO 9 which was modeled both as a steady state and transient (time-limited) release. As suggested by the Shell Maplin Sands LNG releases (Blackmore et al., 1982), the liquid source diameter was determined using a boiling rate of  $0.085 \text{ kg/m}^2 \text{ s}$  for LNG on water.

TABLE IV.1  
SUMMARY OF BURRO9 TEST CONDITIONS  
USED IN EXAMPLE SIMULATIONS

---

Source Rate:	130.0 kg/s
Source Radius:	22.06 m
Wind Speed:	6.5 m/s at 8.0 m
Atmospheric Stability:	C (Pasquill)
Monin-Obukhov Length:	-140. m
Surface Roughness:	$2.05 \times 10^{-4} \text{ m}$
Air Temperature:	$35.4^{\circ} \text{ C}$
Atmospheric Humidity:	12.5%
Surface Temperature:	310 K

---



#### IV.1. Example Input Sessions

The input procedures for simulation of the transient release (RUNNAME-BURRO9) and the steady state release (RUNNAME-BURRO9S) are very similar. Therefore, only the specification of the source rate and extent have been included for the transient release. In the point-by-point discussion of the input procedure, note the following:

- (\*) A line terminator (normally a carriage return) must end every line entered by the user.
- (\*) The file name specification RUNNAME must satisfy system restrictions.
- (\*) When DEGADISIN requests the user to choose an option, all acceptable responses are a single character (capital or lower case). The default responses are denoted by a capital letter inside angle brackets (e.g. <N>). When applicable, a menu of responses is included inside parentheses.
- (\*) For numerical responses, a comma, space, tab, or line terminator (carriage return) may separate the numbers.
- (\*) When a file is used as input (i.e. for the density or transient source input), DEGADISIN reads the same information from the file which would be entered at the terminal in the same order and in the same format.

## NOTES ON STEADY STATE SIMULATION OF BURRO9

- ① Begin the input procedure by execution of DEGADISIN.
- ② The file name specification must follow system restrictions. The DEGADIS model uses this file name along with various file extensions for input and output.
- ③ The Title Block is used to carry any desired comments such as information on the specification of certain parameters.
- ④ The wind field parameters include the wind velocity (m/s) at a specified height (m) and the surface roughness (m).
- ⑤ The Pasquill stability class is used to generate estimates of other atmospheric parameters which follow.
- ⑥ The current settings of pertinent atmospheric parameters are displayed in this list. If any of these are to be changed, the first letter of the parameter to be changed is entered. Note that the default--indicated by <N>--is No for no changes.
- ⑦ The Monin-Obukhov length (Length in the list) is to be changed, so L is entered in response to the prompt.
- ⑧ The list is redisplayed to verify the change and to request any further changes. The (default) response of No causes the program to go to the next question.
- ⑨ The ambient temperature and pressure are entered.
- ⑩ DEGADISIN calculates the ambient air density for the given input parameters.

- ① `% run degadisn`
- Dense Gas Dispersion Model input module.
- ② Enter the simulation name : [DIR]RUNNAME burro9s  
INPUT MODULE — DEGADIS MODEL
- \*\*\*\*\*  
③ Enter Title Block — up to 4 lines of 80 characters  
To stop, type '///'  
Steady state simulation of BURRO 9  
//
- ④ ENTER WIND PARAMETERS — U0 (m/s), Z0 (m), and ZR(m)  
U0 — Wind velocity at reference height Z0  
ZR — Surface Roughness  
6.5,8.,2.05e-4
- ⑤ Enter the Pasquill stability class: (A,B,C,D,E,F) <D> c
- ⑥ The values for the atmospheric parameters are set as follows:  
DELTA: 0.2000  
BETA: 0.9000  
Monin-Obukhov length: -9.3344 m  
Sigma X Coefficient: 0.0200  
Sigma X Power: 1.2200  
Sigma X Minimum Distance: 130.0000 m  
⑦ Do you wish to change any of these?  
(No,Delta,Beta,Length,Coefficient,Power,Minimum) <N> L  
Note: For infinity, ML = 0.0  
Enter the desired Monin-Obukhov length: (m) -140.
- ⑧ The values for the atmospheric parameters are set as follows:  
DELTA: 0.2000  
BETA: 0.9000  
Monin-Obukhov length: -140.0000 m  
Sigma X Coefficient: 0.0200  
Sigma X Power: 1.2200  
Sigma X Minimum Distance: 130.0000 m  
Do you wish to change any of these?  
(No,Delta,Beta,Length,Coefficient,Power,Minimum) <N>
- ⑨ Enter the ambient temperature(C) and pressure(ata): 35.4,0.94  
  
The ambient humidity can be entered as Relative or Absolute.  
Enter either R or A (R or a):  
Enter the relative humidity (Z): 12.5
- ⑩ Ambient Air density is 1.0731 kg/m<sup>3</sup>

- (11) If the release is isothermal, respond "Y". A positive response causes DEGADISIN to ask for a list of concentration, density, and mole fraction points for the gas mixture. The default response is negative.
- (12) If the release is simulated as adiabatic, the default negative response is chosen. For inclusion of heat transfer effects, the surface temperature and the method of calculating the heat transfer coefficient must be specified.
- (13) Water transfer to the source blanket (if present) can be included in the calculation.
- (14) Enter the three-letter designation of the diffusing gas. The properties of LNG as methane and LPG as propane are included.
- (15) A list of the properties for the specified gas (if available) is given. If any of the parameters are to be changed, the first letter of the parameter to be changed in the list is given to the prompt. Here, the level at which the flammability contours are calculated is changed from 0.5 m to 1.0 m.
- (16) The gas property list is displayed again. The default response is no change.
- (17) The lowest concentration of interest is the concentration at which the calculations are stopped.

- (11) Is this an Isothermal spill? <Y or N>
- (12) Is heat transfer to be included in the calculations <Y or N> Y  
 Enter the surface temperature [=] K : 310.  
 Do you want to use the built in correlation, the LLNL correlation, or  
 enter a particular value?  
 (Corr,LLNLcorr,Value) <C>
- (13) Is water transfer to be included in the source <Y or N>
- (14) Enter the code name of the diffusins species: lns
- (15) The characteristics for the gas are set as follows:  
 Molecular weight: 16.04  
 Storage temperature [K]: 111.70  
 Density at storage temperature, PAMB [ks/m<sup>3</sup>]: 1.6845  
 Mean Heat capacity constant 5.60000E-08  
 Mean Heat capacity power 5.0000  
 Upper Flammability Limit [mole frac] 0.15000  
 Lower Flammability Limit [mole frac] 5.00000E-02  
 Height of Flammability Limit [m] 0.50000  
 Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z) <N> z  
 Enter the desired Height for the flammable limit calculations: 1.0
- (16) The characteristics for the gas are set as follows:  
 Molecular weight: 16.04  
 Storage temperature [K]: 111.70  
 Density at storage temperature, PAMB [ks/m<sup>3</sup>]: 1.6845  
 Mean Heat capacity constant 5.60000E-08  
 Mean Heat capacity power 5.0000  
 Upper Flammability Limit [mole frac] 0.15000  
 Lower Flammability Limit [mole frac] 5.00000E-02  
 Height of Flammability Limit [m] 1.0000
- (17) Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z) <N>  
 Enter the LOWEST CONCENTRATION OF INTEREST (ks/m<sup>3</sup>) : 5.e-3

- (18) A initial mass of gas can be specified over the source. This can be used to model aboveground release such as the Thorney Island Trials and should be zero for steady state releases.
- (19) If a steady state release is to be simulated, type "Y" to the prompt. For a steady simulation, the steady state mass evolution rate (kg/s) and primary source extent (m) are required.
- (20) A note about the numerical parameter files is included. These files contain various constant values used in the programs to which the user has access without recompiling the programs. Access is granted as a convenience.
- (21) DEGADISIN will generate a command procedure suitable for running the model under VMS.
- (22) If so desired, DEGADISIN will initiate the command procedure under VMS. If not, the program returns to the operating system.

## Specification of source rate and extent.

(18) Enter the initial mass of pure gas over the source. (kg)  
(Positive or zero): 0.

(19) Is this a Steady state simulation? <y or N> y

Enter the desired evolution rate [=] kg/sec : 130.  
Enter the desired source radius [=] m : 22.06

(20) In addition to the information just obtained, DEGADIS  
requires a series of numerical parameter files which use  
the same name as [DIR]RUNNAME given above.

For convenience, example parameter files are included for  
each step. They are:

EXAMPLE.ER1 and  
EXAMPLE.ER2

Note that each of these files can be edited during the course of the  
simulation if a parameter proves to be out of specification.

(21) Do you want a command file to be generated to execute the procedure? <Y or n>  
The command file will be generated under the file name:  
burro9s.com

(22) Do you wish to initiate this procedure? <y or N>  
y

## NOTES ON TRANSIENT SIMULATION OF BURRO 9

Beginning with the specification of the source rate and extent the responses to all of the previous questions except the simulation name (RUNNAME) are the same for the steady state case and are not repeated.

- ②3 An initial mass of gas can be specified over the source. This can be used to model aboveground releases such as the Thorney Island Trials.
- ②4 The default response is for a transient release.
- ②5 The transient source description consists of ordered triples of time, evolution rate, and source radius.
- ②6 An input file can be used to enter the data triples to avoid typing errors or to use as output from another model such as a liquid spreading model. The file format is the same as the terminal entry format.
- ②7 The first item is the number of triples used in the description followed by the triples with the last two values showing no gas present.
- ②8 DEGADISIN will generate a command procedure suitable for running the model under VMS.
- ②9 If so desired, DEGADISIN will initiate the procedure under VMS. If not, the program returns to the operating system.



Specification of source rate and extent.

- (23) Enter the initial mass of pure gas over the source. (kg)  
(Positive or zero): 0.

- (24) Is this a Steady state simulation? <Y or N>

#### Source Description

- (25) The same form used by the density description  
is used by the source description as follows

first point — time=0 E,R1 at initial (nonzero) values

.

next to last point — time=TEND E,R1=0.

last point time=TEND+ E,R1=0.

- (26) Note: the final time is the last time entered where E and R1 are non-zero  
Do you have an input file for the Source Description? [Y or N]

- (27) ENTER THE NUMBER OF TRIPLES (max= 30) FOR THE SOURCE DESCRIPTION: 4

Enter TIME (sec), EVOLUTION RATE (kg/m<sup>3</sup>s), and POOL RADIUS (m)

0.,130.,22.06

80.,130.,22.06

81.,0.,0.

82.,0.,0.

In addition to the information just obtained, DEGADIS  
requires a series of numerical parameter files which use  
the same name as DIRJRUNNAME given above.

For convenience, example parameter files are included for  
each step. They are:

EXAMPLE.ER1,

EXAMPLE.ER2, and

EXAMPLE.ER3

Note that each of these files can be edited during the course of the  
simulation if a parameter proves to be out of specification.

- (28) Do you want a command file to be generated to execute the procedure? <Y or n>  
The command file will be generated under the file name:  
burro9.com

- (29) Do you wish to initiate this procedure? <Y or N>  
1

The generated INP files for BURRO9S and BURRO9 are shown in Figures IV.1 and IV.2. If necessary, the user may edit the INP file before beginning the simulation. The generated command procedures are shown in Figures II.1 and II.2.

#### IV.2. Example Simulation Output

After proper completion of the model, BURRO9.LIS and BURRO9S.LIS contain the output listing for the transient and steady state releases, respectively. Figure IV.3 shows the predicted maximum ground level centerline concentration for BURRO9 as well as the maximum reported concentration as a function of downwind distance.

## Steady state simulation of BURRO 9

```

6.500000      8.000000      2.050000E-04
3
0.2000000    0.9000000    -140.0000
2.0000000E-02  1.220000    130.0000
308.5500     0.9400000    5.1530441E-03
0 310.0000
1 0.0000000E+00
0 0.0000000E+00
Ins
16.04000     111.7000     1.684480
5.6000001E-08 5.000000
0.1500000    5.0000001E-02    1.000000
4.9999999E-03
0.0000000E+00
4
0.0000000E+00 130.0000     22.06000
6023.000     130.0000     22.06000
6024.000     0.0000000E+00 0.0000000E+00
6025.000     0.0000000E+00 0.0000000E+00
F F F F T F
16-MAY-1985 04:43:28.92
130.0000     39.10033     39.10033

```

Figure IV.1. BURRO9S.INP listing.

## Transient simulation of BURRO 9

```

6.500000      8.000000      2.0500000E-04
3
0.2000000      0.9000000      -140.0000
2.0000000E-02  1.220000      130.0000
308.5500      0.9400000      5.1530441E-03
0  310.0000
1  0.0000000E+00
0  0.0000000E+00
Ins
16.04000      111.7000      1.684480
5.6000001E-08  5.000000
0.1500000      5.0000001E-02  1.000000
4.9999999E-03
0.0000000E+00
4
0.0000000E+00  130.0000      22.06000
80.00000      130.0000      22.06000
81.00000      0.0000000E+00  0.0000000E+00
82.00000      0.0000000E+00  0.0000000E+00
F F F F F
16-MAY-1985 04:37:58.67

```

Figure IV.2. BURRO9.INP listing.

## TEST: Burro 9

## Source Description

Type: Steady, Time-Limited LNG  
 Primary Source Radius (m): 22.1  
 Primary Source Flux ( $\text{kg/m}^2 \text{ s}$ ): 0.085  
 Rate ( $\text{kg/s}$ ): 130.0  
 Temperature: 112

## Meteorological Conditions

Wind Velocity (m/s): 6.5  
 @ Height (m): 8.0  
 Surface Roughness (m): 2.05 E-4  
 Pasquill Stability: C  
 Monin-Obukhov Length (m): -140.0  
 Air Temperature (K): 308.55  
 Relative Humidity (%): 13  
 Surface Temperature (K): 310.0

## Release Richardson Number

Volumetric Release Rate ( $\text{m}^3/\text{s}$ ): 76.9  
 Characteristic Width (m): 39.1  
 Estimated Friction Velocity (m/s): 0.219

$$Ri_0^C = g \left( \frac{\rho_l - \rho_a}{\rho_a} \right) \frac{Q}{u_*^2 D} : 36$$

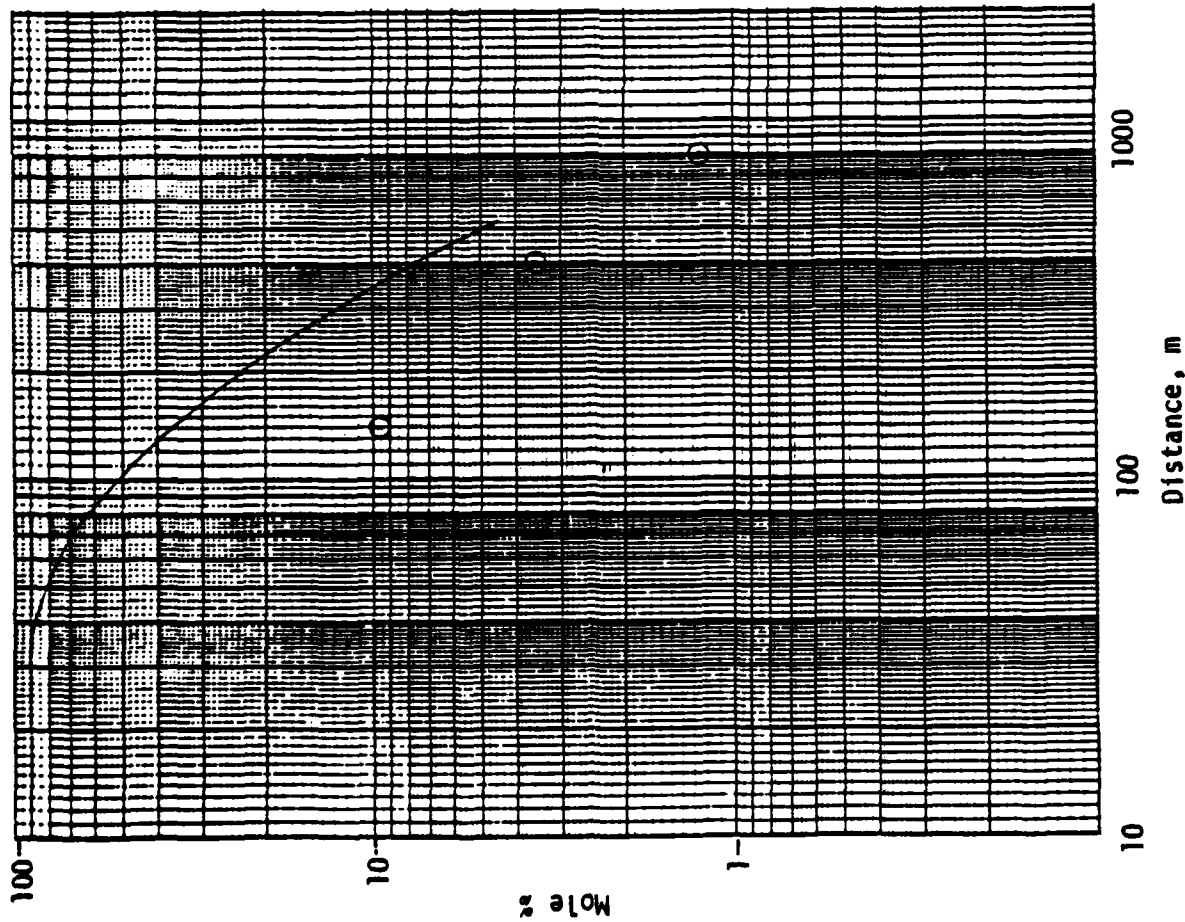


Figure IV.3. DEGADIS-predicted centerline maximum concentration vs. maximum reported concentration--BURRO9.

## REFERENCES

- Beals, G. A., "A Guide to Local Dispersion of Air Pollutants," Air Weather Service Technical Report 214, April, 1971.
- Blackmore, D. R. et al., "Dispersion and Combustion Behavior of Gas Clouds Resulting from Large Spillages of LNG and LPG onto the Sea," Transactions, Institution of Marine Engineers, 94, 1982.
- Businger, J. A., J. C. Wyngaard, Y. Izumi, and E. F. Bradley, "Flux-Profile Relationships in the Atmospheric Surface Layer," Journal of the Atmospheric Sciences, 2, March, 1971.
- Colenbrander, G. W., "A Mathematical Model for the Transient Behavior of Dense Vapor Clouds," 3rd International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Basel, Switzerland, 1980.
- Colenbrander, G. W. and J. S. Puttock, "Dense Gas Dispersion Behavior: Experimental Observations and Model Developments," International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Harrogate, England, September, 1983.
- Koopman, R. P. et al., "Burro Series Data Reports, LLNL/NWC 1980 LNG Spill Tests," Lawrence Livermore National Laboratories Report UCID-19075, December, 1982.
- Pasquill, F., Atmospheric Diffusion, 2nd edition, Halstead Press, New York, 1974.
- van Ulden, A. P., "A New Bulk Model for Dense Gas Dispersion: Two-Dimensional Spread in Still Air," I.U.T.A.M. Symposium on Atmospheric Dispersion of Heavy Gases and Small Particles, Delft University of Technology, The Netherlands, August 29-September 2, 1983.

## APPENDIX A

### DEGADIS MODEL INSTALLATION ON VAX/VMS

DEGADIS was developed under VAX/VMS V3.5 and VAX-11 Fortran V3.5 although there should be no installation difficulty for VAX/VMS V3.2 or later.

The directory which contains the Fortran source code for DEGADIS must be equivalenced with the logical name SYS\$DEGADIS:. If the full directory specification is DQAO:[HACS.DEGADIS], issue the VAX/VMS command:

```
$ ASSIGN DQAO:[HACS.DEGADIS] SYS$DEGADIS:
```

with either the /PROCESS, /GROUP, or /SYSTEM qualifier (/SYSTEM is recommended). Once this assignment is made, the files must be compiled and linked to form DEGADISIN, DEGADIS1, DEGADIS2, DEGADIS3, and SDEGADIS2. The process which compiles and links DEGADIS must have READ, WRITE, and EXECUTE access privileges to SYS\$DEGADIS while only READ and EXECUTE access privileges are needed to execute the existing models.

## APPENDIX B

CONSIDERATIONS FOR INSTALLATION OTHER  
THAN VAX/VMS

There are two types of problems which may occur when attempting to install DEGADIS on a different computer or operating system. The first source of difficulty is the use of non-standard ANSI Fortran 77 language elements. The second source of difficulty is the use of external VAX/VMS routines in DEGADIS.

The following list is a collection of the VAX-11 Fortran extensions which have been used in DEGADIS:

- (\*) In-line comments--An exclamation mark (!) is used to include comments at the end of a valid statement.
- (\*) Special characters--The underscore (\_) is used in variable names.
- (\*) DO loops--DO loops are used with the structure:

```
DO v = e1,e2[,e3]
```

```
.
```

```
.
```

```
.
```

```
END DO
```

where v is a variable name and e1, e2, and e3 are numeric expressions. The numeric expressions have the standard Fortran 77 meaning.

- (\*) INCLUDE statements--INCLUDE statements simply allow other source files to be inserted in the routine being compiled at this point in the source. The system



table '(\$SSDEF)' is used to check the status of returning system routines.

- (\*) OPEN keyword NAME--The OPEN keyword NAME specifies the file name to be opened.
- (\*) Format descriptors--The Q descriptor obtains the integer number of characters remaining in the input record during a READ operation. The dollar sign (\$) descriptor suppresses the carriage return at the end of a line on output.
- (\*) Continuation lines--Continuation lines have been expressed by using either a non-blank character in column 6 or by beginning the line with a tab and a number in the next column.
- (\*) Concatenation of character strings--Character strings are concatenated using two slashes (//).

The following VAX/VMS subroutines have been used in

DEGADIS:

- (\*) SECNDS  
TIME = SECNDS(TIMEO)  
SECNDS returns to TIME the difference between the number of seconds after midnight on the system clock and the value of TIMEO.
- (\*) LIB\$DATE\_TIME  
ISTAT = LIB\$DATE\_TIME (STRING)  
LIB\$DATE\_TIME returns a 24-character ASCII string with the system date and time. ISTAT is an integer variable which accepts the return status.
- (\*) LIB\$DO\_COMMAND  
ISTAT = LIB\$DO\_COMMAND (STRING)  
LIB\$DO\_COMMAND issues the command STRING (a character string) to VAX/VMS. If the command is issued, the calling process is terminated, and ISTAT will contain a failure code.

APPENDIX C  
CODE LISTING

AFGEN.FOR	C-2	RKGST.FOR	C-84
ALPH.FOR	C-3	RTMI.FOR	C-92
CRFG.FOR	C-6	SDEGADIS2.FOR	C-96
DEGADIS1.DEC	C-11	SORTS.FOR	C-104
DEGADIS1.FOR	C-12	SORTS1.FOR	C-107
DEGADIS2.DEC	C-19	SRC1.FOR	C-111
DEGADIS2.FOR	C-20	SRTOUT.FOR	C-119
DEGADIS3.DEC	C-24	SSG.FOR	C-122
DEGADIS3.FOR	C-25	SSGOUT.FOR	C-124
DEGADISIN.DEC	C-29	SSGOUTSS.FOR	C-127
DEGADISIN.FOR	C-30	SSOUT.FOR	C-130
ESTRT1.FOR	C-35	SSSUP.FOR	C-132
ESTRT2.FOR	C-39	STRT2.FOR	C-140
ESTRT2SS.FOR	C-41	STRT2SS.FOR	C-143
ESTRT3.FOR	C-43	STRT3.FOR	C-145
GAMMA.FOR	C-44	SURFACE.FOR	C-147
GETTIM.FOR	C-46	SZF.FOR	C-149
HEAD.FOR	C-48	TPROP.FOR	C-152
ID.FOR	C-53	TRANS1.FOR	C-166
IOT.FOR	C-55	TRANS2.FOR	C-169
NOBL.FOR	C-67	TRANS2SS.FOR	C-171
OB.FOR	C-69	TRANS3.FOR	C-173
PSIF.FOR	C-71	TRAP.FOR	C-174
PSS.FOR	C-72	TS.FOR	C-177
PSSOUT.FOR	C-75	TUPF.FOR	C-178
PSSOUTSS.FOR	C-78	UIT.FOR	C-182
RIPHIF.FOR	C-81		

```

C.....
C
C   THIS FUNCTION LINEARLY INTERPOLATES FROM THE GIVEN
C   PAIR OF DATA POINTS.
C
C   FUNCTION AFGEN(TAB,X,SPEC)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sys$desadis:DEGADIS1.dec'
C   common/nend/poundn,pound
C
C   character*4 pound
C   character*(*) SPEC
C   DIMENSION TAB(1)
C
C   IF(X .GE. TAB(1)) GO TO 95
C   WRITE(1unlog,1100) x,spec
C   AFGEN = TAB(2)
C   RETURN
C
C 95  continue
C     ix = 1
C 100 ix = ix + 2
C
C     iy = ix + 1
C     IF( TAB(ix).EQ.POUNDN .AND. TAB(iy).EQ.POUNDN ) GO TO 500
C     IF(X .GE. TAB(ix)) GO TO 100
C
C     ixp = ix-2
C     iyp = ixp + 1
C
C     SL = (TAB(iy) - TAB(iyp))/(TAB(ix) - TAB(ixp))
C     AFGEN = SL*(X - TAB(ixp)) + TAB(iyp)
C     RETURN
C
C 500 CONTINUE
C     ix = ix-2
C     iy = iy-2
C     ixp = ix-2
C     iyp = iy-2
C
C     SL = (TAB(iy) - TAB(iyp))/(TAB(ix) - TAB(ixp))
C     AFGEN = SL*(X - TAB(ixp)) + TAB(iyp)
C
C 1100 FORMAT(2X,'?AFGEN? UNDERFLOW; argument: ',1p13.5,5X,A20)
C     RETURN
C     END
####

```

```

1 -- sys$desadis:AFGEN.FOR

```

```

C.....
C
C   SUBROUTINES TO CALCULATE THE VALUE OF ALPHA
C
C   SUBROUTINE ALPH
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sys$desadis:DEGADIS1.dec'
C
C   COMMON
C   $/ERROR/STPIN,ERBND,STPMX,WTRG,WTm,WTya,wtyc,wtb,wtah,wtuh,XLI,
C   $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srcut,
C   $ hrcut,ERNOBL,NOBL,et,crfser,epsilon
C   $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/ALP/ALPHA,alpha1
C   $/alphcom/ ialpfl,alpco
C
C   REAL*8 ML,K
C
C   EXTERNAL ALPHI
C
C   PSI = PSIF(ZO,ML)
C   USTAR = UO*K/(dLOG((ZO+ZR)/ZR) - PSI)
C
C   if(u0 .eq. 0.) then
C       alpha = 0.
C       ustar = 0.
C       return
C   endif
C
C   if(ialpfl.eq. 0) then
C       alpha = alpco
C       return
C   endif
C
C*** RMTI USED TO DETERMINE THE ROOT OF THE REQUIRED INTEGRAL EQUATION
C
C   IEND = 40
C   IER = 0
C
C   CALL RMTI(X,F,ALPHI,XLI,XRI,EPS,IEND,IER)
C   IF( IER .NE. 0 ) CALL tran(19,IER)
C   ALPHA = X
C
C   RETURN
C   END
C
C.....
1 -- sys$desadis:ALPH.FOR

```

```

C
C      FUNCTION TO EVALUATE THE WEIGHTED EUCLIDEAN NORM OF THE
C      ERROR ASSOCIATED WITH THE POWER LAW FIT OF THE WIND PROFILE.
C
C      FUNCTION ALPHI(X)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'sys$desadis:DEGADIS1.dec'
C
C      COMMON
C      $/ERROR/STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtyc,wtab,wtab,wtab,XLI,
C      $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srccs,srccut,
C      $ htcut,ERNOBL,NOBLpt,crfser,ersilon
C      $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/ALP/ALPHA,alpha1
C
C      REAL*8 ML,K
C
C      DIMENSION Y(1),DERY(1),PRMT(5),AUX(8)
C      EXTERNAL ARG,ARGOUT
C
C      ALPHA = X
C
C      PRMT(1) = Z0
C      PRMT(2) = dmax1(ZLOW,zr)      ! to take care of large zr
C      PRMT(3) = STPINZ
C      PRMT(4) = ERBNDZ
C      PRMT(5) = STPMXZ
C
C      Y(1) = 0.0D00
C
C      DERY(1) = 1.0D00
C
C      NDIM = 1
C      IHLF = 0
C
C      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,ARG,ARGOUT,AUX)
C
C      IF(IHLF .GE. 10) CALL trap(18,IHLF)
C      ALPHI = Y(1)
C      RETURN
C      END
C
C
C.....
C
C      FUNCTION TO EVALUATE THE ARGUMENT OF THE INTEGRAL EXPRESSION
C
C      SUBROUTINE ARG(Z,Y,D,PRMT)
C
C      2 -- sys$desadis:ALPH.FOR

```

```

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$desadis:DEGADIS1.dec'
C
      COMMON
      $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
      $/ALP/ALPHA,alpha1
      $/alphcom/ ialpfl,alpco
C
      REAL*8 ML,K
C
      DIMENSION Y(1),D(1),PRMT(1)
C
C*** WEIGHT FUNCTION USED
C
      W = 1.D00/(1.D00 + Z)
      if(ialpfl.eq. 2) w= 1.D00
C
C*** WIND VELOCITY @ Z -- BEST FIT
C
      UBST = USTAR/K*(dLOG((Z+ZR)/ZR) - PSIF(Z,ML))
C
C*** WIND VELOCITY @ Z -- POWER LAW APPROXIMATION
C
      UALP = U0 * (Z/Z0) ** ALPHA
C
      D(1) = W * (UBST - UALP) * dLOG(Z/Z0) * UALP
      RETURN
      END
C
C
      SUBROUTINE ARGOUT
      RETURN
      END
****

```

```

C.....
C
C   SUBROUTINE TO CREATE RADG,QSTR,srcden,srcwc,srcwa,srcenth DATA LISTS
C
C   PARAMETERS -- TABLE - WORKSPACE VECTOR
C                  NTAB - DIMENSION OF TABLE DIVIDED BY iout_src
C                  RER - RELATIVE ERROR BOUND OF CREATED
C                        DATA PAIRS BY LINEAR INTERPOLATION
C
C   VALUES OF TIME, RADG, height, QSTR, SZO, wc, wa, rho, Ri,
C                  wc,wa,enthalpy,temp
C   ARE READ INTO
C   TABLE(1) TO TABLE(13) RESPECTIVELY.
C.....
C
C   SUBROUTINE CRFG(TABLE,NTAB,rer)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   DIMENSION TABLE(1)
C
C   include 'sys$desadis:DEGADIS1.dec'
C   parameter (zero= 1.e-20)
C
C   COMMON
C   $/GEN3/ rads(2,max1),qstr(2,max1),srcden(2,max1),srcwc(2,max1),
C   $ srcwa(2,max1),srcenth(2,max1)
C   $/comata/ istab,tamb,pamb,humid,isoft,tsurf,ihtfl,htco,iutfl,wtco
C   $/PARMSC/ RM,SZN,EMAX,RMAX,TSC1,ALEPH,TEND
C   $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   $/NEND/ POUNDN,POUND
C
C   character*4 pound
C
C   LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C   DATA NI/1/
C
C   data iti/1/      ! time - element no 1 in record
C   data irs/2/      ! Rads - element no 2 in record
C   data iqs/4/      ! Qstar - element no 4 in record
C   data idn/8/      ! rho - element no 8 in record
C   data iwc/10/     ! wc - element no 10 in record
C   data iwa/11/     ! wa - element no 11 in record
C   data ien/12/     ! enthalpy - element no 12 in record
C
C
C   OUTPUT CREATED VECTORS TO A PRINT FILE
C
C
1 -- sys$desadis:CRFG.FOR

```

```

C
  READ(9,*) (TABLE(J),J=1,iout_src)
C
  WRITE(8,1111)
  WRITE(8,1105)
  if(isof1.eq. 1) then
    WRITE(8,1102)
    WRITE(8,1103)
    WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
  else
    WRITE(8,1100)
    WRITE(8,1104)
    WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
  endif
  ispace = 1
C
1100  FORMAT(/,5X,'Time',5X,2X,'Gas Radius',2X,4X,'Height',4X,
  $4X,'Qstar',5X,2X,'SZ(x=L/2.)',2X,1X,'Mole frac C',2X,
  $3X,'Density',4X,1X,'Temperature',2X,3X,'Rich No.',3X)
1102  FORMAT(/,5X,'Time',5X,2X,'Gas Radius',2X,4X,'Height',4X,
  $4X,'Qstar',5X,2X,'SZ(x=L/2.)',2X,1X,'Mole frac C',2X,
  $3X,'Density',4X,3X,'Rich No.',3X)
1103  FORMAT(1H ,5X,'sec',6X,6X,'m',7X,6X,'m',7X,
  $2X,'kg/m**2/s',3X,6X,'m',7X,14X,3X,'kg/m**3',4X,14X,/)
1104  FORMAT(1H ,5X,'sec',6X,6X,'m',7X,6X,'m',7X,
  $2X,'kg/m**2/s',3X,6X,'m',7X,14X,3X,'kg/m**3',4X,6X,'K',7X,14X,/)
1105  FORMAT(1H ,23X,'*****',21X,'CALCULATED SOURCE PARAMETERS',21X,
  $'*****')
C
  RADG(1,1) = 0.
  RADG(2,1) = TABLE(2)
  QSTR(1,1) = 0.
  QSTR(2,1) = TABLE(4)
  srcden(1,1) = 0.
  srcden(2,1) = table(8)
  srcwc(1,1) = 0.
  srcwc(2,1) = table(10)
  srcwa(1,1) = 0.
  srcwa(2,1) = table(11)
  srcenth(1,1) = 0.
  srcenth(2,1) = table(12)
C
  READ(9,*) (TABLE(J),J=1,iout_src)
  L = 2
C
C*** L IS THE NUMBER OF RECORDS WHICH HAVE BEEN READ
C
100  CONTINUE
    DO 120 I=2,NTAB
C
C*** MOVE LAST RECORD READ INTO THE LAST ACTIVE POSITION OF TABLE
2 -- sys$desadis:CRFG.FOR

```



```

C
DO 130 J = 1,iout_src
  KK = iout_src * (I-1) + J
130  TABLE(KK) = TABLE(J)
  KK = iout_src * I
C
C*** READ THE NEXT RECORD. INCREMENT L.
C
  L = L + 1
  READ(9,*,END=900) (TABLE(J),J=1,iout_src)
C
C
DO 140 Kkk = 2,I
C
  KT   = iout_src*(Kkk-1) + iti
  KRG  = iout_src*(Kkk-1) + irg
  KQSTR = iout_src*(Kkk-1) + ias
  KCA  = iout_src*(Kkk-1) + idn
  Kwc  = iout_src*(Kkk-1) + iwc
  Kwu  = iout_src*(Kkk-1) + iwa
  Ken  = iout_src*(Kkk-1) + ien
C
  timeslot = rads(1,ni)
  ratio = (table(kt) - timeslot) / (table(iti) - timeslot)
C
  ANSRG = (TABLE(irs) - RADG(2,NI)) * ratio + RADG(2,NI)
  ANSQ  = (TABLE(ias) - QSTR(2,NI)) * ratio + QSTR(2,NI)
  ANSCA = (TABLE(idn) - srcden(2,NI)) * ratio + srcden(2,NI)
  ANSwC = (TABLE(iwc) - srcwc(2,NI)) * ratio + srcwc(2,NI)
  ANSwa = (TABLE(iwa) - srcwa(2,NI)) * ratio + srcwa(2,NI)
  ANSen = (TABLE(ien) - srcenth(2,NI)) * ratio + srcenth(2,NI)
C
  ERRG = ABS(ANSRG - TABLE(KRG))/TABLE(KRG)
  ERQSTR = ABS(ANSQ - TABLE(KQSTR))/(TABLE(KQSTR)+zero)
  ERO = dMAX1(ERRG,ERQSTR)
C
  ERCA = ABS(ANSCA - TABLE(KCA))/TABLE(KCA)
  ERO = dMAX1(ERO,ERCA)
C
  ERwC = ABS(ANSwC - TABLE(KwC))/(TABLE(KwC)+ zero)
  ERO = dMAX1(ERO,ERwC)
  ERwa = ABS(ANSwa - TABLE(Kwa))/(TABLE(Kwa)+ zero)
  ERO = dMAX1(ERO,ERwa)
  ERen = ABS(ANSen - TABLE(Ken))/(TABLE(Ken)+ zero)
  ERO = dMAX1(ERO,ERen)
C
  IF(ERO .GT. RER) GO TO 150
140  CONTINUE
120  CONTINUE
C
  WRITE(lunlog,1110)

3 -- sys$desadis:CRFG.FOR

```

```

C
C*** RECORD NEXT DATA PAIR. SINCE ERROR EXCEEDED, RECORD THE LAST
C*** DATA PAIR WHICH SATISFIED THE ERROR CRITERIA WHICH IS STORED
C*** IN TABLE(KK-(iout_src-1)) TO TABLE(KK)

```

```

C
150 NI = NI + 1
    IF(NI .GT. MAXL)CALL trap(5)

```

```

C
    KT   = KK - iout_src + iti
    KRG  = KK - iout_src + irg
    KQSTR = KK - iout_src + ias
    KCA  = KK - iout_src + idn
    KWC  = KK - iout_src + iwc
    Kwa  = KK - iout_src + iwa
    Ken  = KK - iout_src + ien

```

```

C
    RADG(1,NI) = TABLE(KT)
    RADG(2,NI) = TABLE(KRG)
    QSTR(1,NI) = TABLE(KT)
    QSTR(2,NI) = TABLE(KQSTR)
    srden(1,NI) = TABLE(KT)
    srden(2,NI) = TABLE(KCA)
    srcwc(1,NI) = TABLE(KT)
    srcwc(2,NI) = TABLE(KWC)
    srcwa(1,NI) = TABLE(KT)
    srcwa(2,NI) = TABLE(KWA)
    srcenth(1,NI) = TABLE(KT)
    srcenth(2,NI) = TABLE(KEN)

```

```

C
C*** WRITE THE POINTS JUST RECORDED TO UNIT=8

```

```

C
    if(isof1.eq. 1) then
        WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),table(kt+8)
    else
        WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),
        1      table(kt+12),table(kt+8)
    endif
    ispace = ispace + 1
    if(ispace.eq. 3) then
        ispace = 0
        write(8,1111)
    endif

```

```

C
    GO TO 100

```

```

C
900 CONTINUE          ! EOF encountered

```

```

C
    NI = NI + 1
    IF(NI+1 .GT. MAXL) CALL trap(5)

```

```

C
    RADG(1,NI) = TABLE(iti)

```

```

4 -- sys$desadis:CRFG.FOR

```

```

RADG(2,NI) = TABLE(irs)
QSTR(1,NI) = TABLE(iti)
QSTR(2,NI) = TABLE(ies)
srcden(1,NI) = TABLE(iti)
srcden(2,NI) = TABLE(idn)
srcwc(1,NI) = TABLE(iti)
srcwc(2,NI) = TABLE(iwc)
srcwa(1,NI) = TABLE(iti)
srcwa(2,NI) = TABLE(iwa)
srcenth(1,NI) = TABLE(iti)
srcenth(2,NI) = TABLE(ien)

```

C

```

if(isof1.eq. 1) then
WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
else
WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
endif
ispace = ispace + 1
if(ispace.eq. 3) then
    ispace = 0
    write(8,1111)
endif

```

C

```

NI = NI + 1
DO 910 I = 1,2
RADG(I,NI) = POUNDN
QSTR(I,NI) = POUNDN
srcden(I,NI) = POUNDN
srcwc(I,NI) = POUNDN
srcwa(I,NI) = POUNDN
srcenth(I,NI) = POUNDN

```

910 CONTINUE

C

RETURN

C

```

1110 FORMAT(' ?CRFG? TABLE exceeded without point selection ',
$'- execution continuing')
1111 FORMAT(1H )
c1140 FORMAT(1H ,<iout_src>(1PG13.6,1X))
1140 FORMAT(1H ,9(1PG13.6,1X))
END

```

####

```

C.....
C
C
C   DIMENSIONS/DECLARATIONS for DEGADIS1
C
C   include 'sys$degadis:DEGADISIN.dec'
C
C   max1 is the length of the /GEN3/ output vectors
C
C   parameter (      lunlog= 6,
1                   sortpi= 1.77 245 3851,      ! sort(pi)
2                   max1= 600,
3                   max12= 2*max1,
4                   iout_src= 13)
C
****

```

## PROGRAM DEGADIS1

C  
 C\*\*\*\*\*  
 C\*\*\*\*\*  
 C\*\*\*\*\*

C  
 C Program description:

C DEGADIS1 estimates the ambient wind profile power alpha and  
 C characterizes the primary gas source.

C  
 C Program usage:

C Consult Volume III of the Final Report to U. S. Coast Guard  
 C contract DT-CG-23-80-C-20029 entitled "Development of an  
 C Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".

C J. A. Havens  
 C T. O. Spicer

C University of Arkansas  
 C 227 Engineering Building  
 C Department of Chemical Engineering  
 C Fayetteville, AR 72701

C April 1985

C This project was sponsored by the U. S. Coast Guard and the Gas  
 C Research Institute under contract DT-CG-23-80-C-20029.

C  
 C Disclaimer:

C This computer code material was prepared by the University of  
 C Arkansas as an account of work sponsored by the U. S. Coast Guard  
 C and the Gas Research Institute. Neither the University of Arkansas,  
 C nor any person acting on its behalf:

- C a. Makes any warranty or representation, express or implied,  
 C with respect to the accuracy, completeness, or usefulness  
 C of the information contained in this computer code material,  
 C or that the use of any apparatus, method, numerical model,  
 C or process disclosed in this computer code material may not  
 C infringe privately owned rights; or
- C b. Assumes any liability with respect to the use of, or for  
 C damages resulting from the use of, any information,  
 C apparatus, method, or process disclosed in this computer  
 C code material.

1 -- sys\$degadis:DEGADIS1.FOR

```

C
C
C*****
C*****
C*****
C
C
C
C      DIMENSIONS/DECLARATIONS
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$desadis:DEGADIS1.dec'
C
C      ntab is the dimension of table divided by iout_src
C
      parameter (      ntab0=910,
$                  ntab=ntab0/iout_src)
C
      include '($sdef)'
C
C      BLOCK COMMON
C
      COMMON
$ /GEN3/  rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
$  srcws(2,max1),srcenth(2,max1)
$ /TITL/  TITLE
$ /GEN1/  ET(2,isen),R1T(2,isen)
$ /GEN2/  DEN(5,isen)
$ /ITI/   T1,TINP,TSRC,TOBS,TSRT
$ /ERROR/STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtyc,wteb,wtab,wtuh,XLI,
$  XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srccs,srccut,
$  htcut,ERNOBL,NOBLat,crfser,epsilon
$ /PARM/  UO,ZO,ZR,HL,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
$ /SZFC/  szstp0,szerr,szstpax,szsz0
$ /com_sprop/  sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpr,
$  sas_ufl,sas_lfl,sas_zsp,sas_name
$ /comata/  istab,tamb,paab, humid, isofl,tsurf,ihtfl,htco,iutfl,wtco
$ /PARMSC/  RM,SZH,EMAX,RMAX,TSC1,ALEPH,TEND
$ /com_ss/  ess,slen,suid,outcc,outcz,outb,outl,swcl,swal,seni,srhl
$ /PHLAG/  CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
$ /vucom/  vuc,vub,vuc,vud,vudelta,vuflas
$ /com_six/  six_coeff,six_pow,six_min_dist,six_flg
$ /com_ENTHAL/  H_masrte,H_airrte,H_watrte
$ /END/  POUNDN,POUND
$ /ALP/  ALPHA,alpha1
$ /alehcom/  ialefl,aleco
$ /phicom/  iphifl,dellay
$ /sard_con/  ce, delrhomin
$ /COM_SURF/  HTCUTS
C
C -- sys$desadis:DEGADIS1.FOR

```

```

C
C      character*80 TITLE(4)
C
C      character*4 pound
C      character*24 TSRC,TINP,TOBS,TSRT
C      character*3 sas_name
C
C      real*4 tt1
C      REAL*8 ML,K
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      logical vuflag
C
C      REAL*8 L,L0
C      DIMENSION PRMT(25),Y(7),DERY(7),AUX(8,7)
C      EXTERNAL SRC1,SRC10
C      character*40 opnrup1
C      character OPNRUP(40)
C      character*4 INP,ER1,SCD,TR2,scl
C
C      dimension table(ntab0)
C
C      equivalence(opnrup(1),opnrup1)
C
C.....
C
C      DATA
C
C      DATA POUNDN/-1.E-20/,POUND/'/'/'/'
C      DATA USTAR/0./,GAMMAF/0./
C      DATA G/9.81/,K/0.35/
C      DATA GAMMAF/0./
C      DATA PRMT/25*0./
C      DATA Y/7*0./,DERY/7*0./
C      DATA TIME0/0./,NDIM/0/
C      DATA EMAX/0./,TSC1/0./
C      DATA ET/isen*0.,isen*0./,R1T/isen*0.,isen*0./
C      data DEN/isen*0.,isen*0.,isen*0,isen*0.,isen*0./
C
C
C      DATA INP/'./,inp'',ER1/'./,er1''
C      DATA SCD/'./,scd'',TR2/'./,tr2''
C      data scl/'./,scl''
C
C.....
C
C      MAIN
C
C*** GET THE EXECUTION TIME
C
C      t1 = secnds(0.)
C      istat = lib$date_TIME(TSRC)
C
C      3 -- sys$degadis:DEGADIS1.FOR

```

```

      if(listat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
C      WRITE(lunlos,1130)
c1130 FORMAT(5X,'Enter the file name beins used for this run: ',%)
      read(5,1000) nchar,opnrup      ! unit 5 sets command file too
1000 format(a,40a1)
C
      opnrup1 = opnrup1(1:nchar) // er1(1:4)
C
      CALL ESTR1(OPNRUP1)
      HTCUTS = HTCUT
C
      opnrup1 = opnrup1(1:nchar) // inp(1:4)
      CALL IO(tend,$mass0,OPNRUP1)
      CALL ALPH
C
      alpha1 = alpha+1.
      WRITE(lunlos,1105) ALPHA
1105 FORMAT(5X,'THE VALUE OF ALPHA IS ',F6.4)
C
C
      GAMMAF = GAMMA(1./ALPHA1)
      TSC1 = TEND
C
C
C*** set the density and enthalpy functions in TPROP
C
      call setenthal(h_masrte,h_airrte,h_watrte)
      call setden(1.D00,0.D00,h_masrte)
C
C.....
C
C      SOURCE INTEGRATION (CA = RHOE)
C
      opnrup1 = opnrup1(1:nchar) // scd(1:4)
C
      OPEN(UNIT=9,NAME=OPNRUP1,recl=202,TYPE='SCRATCH',
$ carriagecontrol='list',
$ recordtype='variable')
C
      $mass = $mass0
C
C.....
C
C      START THE GAS BLANKET?
C
      L = SQRTPI*AFGEN(R1T,TIME0,'R1T-MN')
      QSTRE = AFGEN(ET,TIME0,'ET-MN')/L/L
C
4 -- sys$desadis:DEGADIS1.FOR

```



```

C
  gstar = 0.
  if(u0 .ne. 0.)
    1 gstar = rhoe*k*ustar*alpha1*dellay/(dellay-1.)/phihat(rhoe,L)
C
C
  write(lunlos,3010) time0,l,sz,gstar,gstre
  IF(QSTRE.lt.QSTaR .and. smass0.eq.0.) then
    tsc1 = 0.
    GOTO 105
  endif
C
  100 CONTINUE
  check3 = .false.
C
C*** INITIAL CONDITIONS
C
  if(time0.ne. 0.) then
    L0 = SQRTPI*AFGEN(R1T,TIME0,'R1T-MN')
    QSTRE = AFGEN(ET,TIME0,'ET-MN')/L0/L0
C
    gstar = 0.
    if(u0.ne. 0.)
      1 gstar = rhoe*k*ustar*alpha1*dellay/(dellay-1.)/phihat(rhoe,L0)
C
    endif
C
C*** SET UP INTEGRATION PARAMETERS
C
C*** VARIABLE      SUBSCRIPT
C*** -----
C*** RG            Y(1)
C*** mass          Y(2)
C*** massc         Y(3)
C*** massa         Y(4)
C*** Enthalpy      Y(5)
C*** moe           y(6)
C*** TIME          X
C
  PRMT(1) = TIME0
  PRMT(2) = 6.023E23
  PRMT(3) = STPIN
  PRMT(4) = ERBND
  PRMT(5) = STPMX
C
  PRMT(6) = EMAX -- OUTPUT
C
  do iii = 6,23
    prmt(iii) = 0.
  enddo
C
  Y(1) = AFGEN(R1T,TIME0,'R1T-MN')
C
5 -- sys3desadis:DEGADIS1.FOR

```

```

Y(2) = dmax1( smass, (pi*Y(1)**2*1.1*srccut*rhoe))
vuflag = .false.
htod = smass/rhoe/2./pi/Y(1)**3
prmt(22) = htod * 2.*Y(1)      ! initial height of the tail
prmt(24) = prmt(22)
prmt(23) = 0.                  ! initial height of the head
prmt(25) = prmt(23)
if(htod .gt. 0.1) vuflag = .true.
Y(3) = Y(2)
Y(4) = 0.0
Y(5) = Y(3) * h_masrte
y(6) = 0.0

C
DERY(1) = WTRG
DERY(2) = WTta
DERY(3) = WTyc
DERY(4) = WTya
DERY(5) = WTab
dery(6) = wtab

C
NDIM = 6

C
C*** PERFORM INTEGRATION
C
WRITE(lunlos,1145)
1145 FORMAT(5X,'Beginning Integration Step - Gas blanket')
C
CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SRC1,SRC10,AUX)
C
IF(IHLF .GT. 10) CALL trap(1,IHLF)
IF(CHECK3) then
    TEND = TSC1
    GO TO 110
end if

C
C.....
C
C
C   RESTART THE GAS BLANKET?
C
TIME0 = TSC1
L = SQRTPI*AFGEN(R1T,TIME0,'R1T-MN')
QSTRE = AFGEN(ET,TIME0,'ET-MN')/L/L

C
astar = 0.
if(u0.ne. 0.)
    1 astar = rhoe*k*ustar*alpha1*dellay/(dellay-1.)/phiat(rhoe,L)

C
write(lunlos,3010) time0,l,sz,astar,astre
3010 format(//,' time0: ',1p13.5,t40,' l: ',1p13.5,
/, ' sz0: ',1p13.5,/, ' astar: ',1p13.5,t40,' astre: ',1p13.5)
IF(QSTRE .GT. QSTAR) GO TO 100

6 -- sysidesadis:DEGADIS1.FOR

```

```

C.....
C
C   SOURCE INTEGRATION -- NO GAS BLANKET
C
  105 continue
      WRITE(lunlos,1146)
  1146 FORMAT(5X,'Source calculation - No Gas blanket')
C
      CALL NOBL(timeout)
C
      if(check3) then          ! restart blanket calculation
          time0 = timeout
          goto 100
      end if
C
C
  110 RMAX = 1.01*RMAX          ! GUARANTEE A GOOD VALUE
      aleph = 0.
      if(u0 .ne. 0.)
          1      ALEPH = U0/GAMMAF*(SZM/Z0)**ALPHA
          2      /(SQRTPI/2.*RM +RMAX)**(ALPHA/ALPHA1)
C
C
      rewind (unit=9)
      opnrup1 = opnrup1(1:nchar) // scl(1:4)
      open(unit=8,name=opnrup1, type='new',
          &      carriagecontrol='fortran', recordtype='variable')
C
      call head(gmass0)
      call crfs(table,ntab,crfser)
      call head(gmass0)
C
      CLOSE(UNIT=9)
      close(unit=8)
C
      opnrup1 = opnrup1(1:nchar) // tr2(1:4)
      CALL TRANS(OPNRUP1)
C
C*** CALCULATE EXECUTION TIME
C
      tt1 = t1
      t1 = Secnds(tt1)/60.
      WRITE(lunlos,2000) TSRC
      WRITE(lunlos,2010) T1
  2000 FORMAT(1X,'BEGAN AT ',A24)
  2010 FORMAT(5X,' ***** ELAPSED TIME ***** ',1P,13.5,' min ')
C
      STOP
      END

```

```

****

```

```

7 -- sys$desadis:DEGADIS1.FOR

```

```

C.....
C
C
C   DIMENSIONS/DECLARATIONS for DEGADIS2
C
C   include 'sys$desadis:DEGADIS1.dec/list'
C
C   MAXNOB IS THE MAXIMUM NUMBER OF OBSERVERS ALLOWED.
C
C   parameter(      maxnob = 50,
1                  RT2= 1.41 421 3562,      ! sart(2.0)
2                  sapio2= 1.25 331 4137) ! sart(pi/2.)
C
####

```

## PROGRAM DEGADIS2

C  
 C\*\*\*\*\*  
 C\*\*\*\*\*  
 C\*\*\*\*\*

C  
 C Program description:

C  
 C DEGADIS2 performs the downwind dispersion portion of the calculation  
 C for each of several observers released successively over the transient  
 C gas source described by DEGADIS1.

C  
 C Program usage:

C  
 C Consult Volume III of the Final Report to U. S. Coast Guard  
 C contract DT-CG-23-80-C-20029 entitled 'Development of an  
 C Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.

C  
 C J. A. Havens  
 C T. O. Spicer

C  
 C University of Arkansas  
 C 227 Engineering Building  
 C Department of Chemical Engineering  
 C Fayetteville, AR 72701

C  
 C April 1985

C  
 C This project was sponsored by the U. S. Coast Guard and the Gas  
 C Research Institute under contract DT-CG-23-80-C-20029.

C  
 C Disclaimer:

C  
 C This computer code material was prepared by the University of  
 C Arkansas as an account of work sponsored by the U. S. Coast Guard  
 C and the Gas Research Institute. Neither the University of Arkansas,  
 C nor any person acting on its behalf:

- C  
 C a. Makes any warranty or representation, express or implied,  
 C with respect to the accuracy, completeness, or usefulness  
 C of the information contained in this computer code material,  
 C or that the use of any apparatus, method, numerical model,  
 C or process disclosed in this computer code material may not  
 C infringe privately owned rights; or  
 C  
 C b. Assumes any liability with respect to the use of, or for  
 C damages resulting from the use of, any information,  
 C apparatus, method, or process disclosed in this computer

```
C      code material.
C
C
C*****
C*****
C*****
C
C
C
C
C
C
C
C
C
DIMENSIONS/DECLARATIONS

Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

include 'sys$degadis:DEGADIS2.dec/list'
include '($ssdef)'

COMMON
$/GEN3/ rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
$ srcws(2,max1),srcenth(2,max1)
$/SSCON/ NREC(maxnob,2),T0(maxnob),XU(maxnob)
$/TITL/ TITLE
$/GEN1/ ET(2,isen),R1T(2,isen)
$/GEN2/ DEN(5,isen)
$/PARM/ UO,ZO,ZR,HL,USTAR,K,G,RHOE,RHQA,DELTA,BETA,GAMMAF,CcLOW
$/com_prop/ gas_mu,gas_temp,gas_rho,gas_cpk,gas_cpp,
$ gas_ufl,gas_lfl,gas_zsp,gas_name
$/ITI/ Tl,TINP,TSRC,TOBS,TSRT
$/ERROR/SYOE,ERRO,SZOER,WTAID,WTQOO,WT SZO,ERRP,SMXP,
$ WTSZP,WT SYP,WTBEP,WT DH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WT DHG
$/comata/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iutfl,wteo
$/PARMSC/ RH,SZH,EMAX,RMAX,TSC1,ALEPH,TEND
$/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
$/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
$/com_sisx/ sisx_coeff,sisx_pow,sisx_min_dist,sisx_flg
$/NEND/ POUNDN,POUND
$/ALP/ ALPHA,alpha1
$/phicom/ iphifl,dellay
$/sprd_con/ ce, delrhomin
$/COM_SURF/ HTCUT
$/STOPIT/ TSTOP
$/CN OBS/ NOBS

character*80 TITLE(4)

character*4 pound
character*24 TINP,TSRC,TOBS,TSRT

2 -- sys$degadis:DEGADIS2.FOR
```

```

C      character*3 sas_name
C      real*4 tt1
C      REAL*8 K,ML,L
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C      character*4 TR2,ER2,PSD,TR3
C      character OPNRUP(40)
C      character*40 OPNRUP1
C      equivalence (opnrup(1),opnrup1)
C
C.....
C
C      DATA
C
C      DATA TSTOP/0./
C      DATA POUND/'/'/'/'/'',POUNDN/-1.E-20/
C
C      DATA TIME0/0./,NDIM/0/
C      DATA RADG/max12*0./,QSTR/max12*0./,srcden/max12*0./
C      DATA NREC/maxnob*0,maxnob*0/,T0/maxnob*0./,XU/maxnob*0./
C
C      DATA TR2/''.TR2'',ER2/''.ER2''
C      DATA PSD/''.PSD'',TR3/''.TR3''
C.....
C
C      MAIN
C
C      T1 = SECNDS(0.)
C      istat = lib$date_TIME(TOBS)
C      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
C      WRITE(5,1130)
C1130 FORMAT(' Enter the file name beins used for this run: ',%)
C      read(5,1130) nchar,opnrup
C      1130 format(a,40a1)
C
C      opnrup1 = OPNRUP1(1:nchar) // er2(1:4)
C      CALL ESTR2(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
C      opnrup1 = OPNRUP1(1:nchar) // tr2(1:4)
C      CALL S1RT2(OPNRUP1,H_msrte)
C.....
C
3 -- sys$degadis:DEGADIS2.FOR

```

```

C   PSEUDO STEADY STATE CALCULATIONS
C   INTEGRATION IN SUBROUTINE SUPERVISOR
C
      OPNRUP1 = OPNRUP1(1:nchar) // PSD(1:4)
      OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1,
        $  carriagecontrol='list',
        $  recordtype='variable')
C
      CALL SSSUP(H_masrte)
C
C.....
C
      CLOSE(UNIT=9)
C
C
      call setden(1.,0.,H_masrte)      ! adiabatic mixing w/ pure stuff
C
C
      OPNRUP1 = OPNRUP1(1:nchar) // tr3(1:4)
      CALL TRANS(OPNRUP1)
C
      tt1 = t1
      T1 = SECNDS(tt1)
      T1 = T1/60.
      WRITE(lunlos,4000) T0BS
      WRITE(lunlos,4010) T1
      4000 FORMAT(3X,'BEGAN AT ',A40)
      4010 FORMAT(3X,'*** ELAPSED TIME *** ',1P613.5,' min')
C
      STOP
      END
****

```



```
C.....  
C  
C   declarations for DEGADIS3  
C  
C   include 'sys$desadis:DEGADIS2.dec/list'  
C  
C   parameter (      maxnt=40,  
C   $               maxtnob=maxnt$maxnob)  
C  
####
```

## PROGRAM DEGADIS3

```

C
C*****
C*****
C*****
C
C   Program description:
C
C   DEGADIS3 sorts the downwind dispersion calculation made for each of
C   the several observers in DEGADIS2. The output concentrations at
C   several given times may then be corrected for along-wind dispersion
C   as desired.
C
C
C   Program usage:
C
C   Consult Volume III of the Final Report to U. S. Coast Guard
C   contract DT-CG-23-80-C-20029 entitled 'Development of an
C   Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.
C
C   J. A. Havens
C   T. O. Spicer
C
C   University of Arkansas
C   227 Engineering Building
C   Department of Chemical Engineering
C   Fayetteville, AR 72701
C
C   April 1985
C
C   This project was sponsored by the U. S. Coast Guard and the Gas
C   Research Institute under contract DT-CG-23-80-C-20029.
C
C
C   Disclaimer:
C
C   This computer code material was prepared by the University of
C   Arkansas as an account of work sponsored by the U. S. Coast Guard
C   and the Gas Research Institute. Neither the University of Arkansas,
C   nor any person acting on its behalf:
C
C   a. Makes any warranty or representation, express or implied,
C      with respect to the accuracy, completeness, or usefulness
C      of the information contained in this computer code material,
C      or that the use of any apparatus, method, numerical model,
C      or process disclosed in this computer code material may not
C      infringe privately owned rights; or
C
C   b. Assumes any liability with respect to the use of, or for
C      damages resulting from the use of, any information;

```

```

1 -- sys$desadis:DEGADIS3.FOR

```

```

C      apparatus, method, or process disclosed in this computer
C      code material.
C
C
C

```

```

C*****
C*****
C*****

```

```

C
C
C
C

```

```

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

```

```

      include 'sws$degadis:DEGADIS3.dec/list'
      include '($ssdef)'

```

```

C

```

```

C*** MINIMUM DIMENSION ON TABLE IS 6 * MAXNOB + 1

```

```

C

```

```

      parameter (ntab0=10*maxnob+1)

```

```

C

```

```

      COMMON

```

```

$/SORT/ TCA(maxnob,maxnt),TCASTR(maxnob,maxnt),
$      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
$      Tsama(maxnob,maxnt),Ttemp(maxnob,maxnt),
$      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
$      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
$/SSCON/ HREC(maxnob,2),T0(maxnob),XV(maxnob)
$/SORTIN/ TIM(maxnt),NTIM,ISTR
$/GEN2/ DEN(5,isen)
$/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CALOW
$/com_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
$ sas_ufl,sas_lfl,sas_zsp,sas_name
$/ITI/ T1,TINP,TSRC,TOBS,TSRT
$/comata/ istab,tamb,pamb,humid,isoft,tsurf,ihtfl,htco,iwtfl,wtco
$/PARMSC/ RM,SM,EMAX,RMAX,TSC1,ALEPH,TEND
$/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
$/com_six/ six_coeff,six_pow,six_min_dist,six_flg
$/ERROR/ ERT1,ERDT,ERNTIM
$/NEND/ POUNDN,POUND
$/ALP/ ALPHA,alpha1
$/CNOBS/ NOBS

```

```

C

```

```

      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      REAL*8 ML,K
      DIMENSION TABLE(ntab0)

```

```

C

```

```

      character*24 tsrc,tinp,tobs,tsrt

```

```

C

```

```

      character*3 sas_name
      character*4 TR3,PSD,Er3,SR3,Tr4

```

```

character#40 opnrup1
character opnrup(40)
C
EQUIVALENCE (OPNRUP(1),opnrup1)
C.....
C
C DATA
C
DATA POUNDN/-1.E-20/,POUND/'/'/'/'
DATA TCA/maxtnob#0./,TCASTR/maxtnob#0./,TSY/maxtnob#0./
data TSZ/maxtnob#0./,KSUB/maxnt#0/
DATA TB/maxtnob#0./,TDIST0/maxtnob#0./,TDIST/maxtnob#0./
C
DATA TR3/''.TR3'/',PSD/''.PSD'/'
DATA Er3/''.Er3'/',SR3/''.SR3'/',Tr4/''.Tr4'/'
C
C*** UNITS
C*** 8 -- OUTPUT TO A PRINT FILE
C*** 9 -- I/O WITH DISK
C
T1 = SECNDS(0.)
istat = lib$date_time(tsrt)
if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c WRITE(5,1130)
c1130 FORMAT(' Enter the file name used for this run: ',%)
read(5,1130) nchar,opnrup
1130 format(a,40a1)
C
C*** GET THE VERSION NUMBER
C
c 100 WRITE(5,1140)
c1140 FORMAT(' Enter the version number (between 00 and 99) for',
c '$' this sort: ',%)
c CALL GTLIN(DUMMY)
c NCAR = LEN(DUMMY)
c IF(NCAR .EQ. 0) GO TO 110
C
c IF(IVERIF(DUMMY,STRING) .NE. 0) GO TO 110
c IF(NCAR-2) 130,140,120
C
c 110 WRITE(5,1150)
c1150 FORMAT(' ?DEGADIS3? - Invalid character for version number')
c GO TO 100
C
c 120 WRITE(5,1160)
c1160 FORMAT(' ?DEGADIS3? - Too many characters in the version number')
c GO TO 100

3 -- sys$degadis:DEGADIS3.FOR

```

```

C
c 130 DOT(1) = '060
c     DOT(2) = DUMMY(1)
c     GO TO 150
C
c 140 DOT(1) = DUMMY(1)
c     DOT(2) = DUMMY(2)
C
c 150 CONTINUE
c     CALL CONCAT(Er3,DOT,Er3)
c     CALL CONCAT(Sr3,DOT,Sr3)
c     CALL CONCAT(Tr4,DOT,Tr4)
C
C*** NOW, REPLACE THE FILE NAME IN OPNRUP
C
c     CALL SCOPY(BFILE,OPNRUP)
C
C*** THATS IT
C
c     OPNRUP1 = OPNRUP1(1:nchar) // tr3(1:4)
c     CALL STRT3(OPNRUP1)
C
c     OPNRUP1 = OPNRUP1(1:nchar) // er3(1:4)
c     CALL ESTRT3(OPNRUP1)
C
c     OPNRUP1 = OPNRUP1(1:nchar) // psd(1:4)
c     OPEN(UNIT=9,NAME=OPNRUP1,TYPE='OLD')
C
C.....
C
C     TIME SORT SUPERVISOR -- CALCULATE DOWNWIND DISPERSION CORRECTION
C
c     CALL SORTS(TABLE)
C
c     CLOSE(UNIT=9)
C
C.....
C
C     OUTPUT SORTED PARAMETERS
C
c     OPNRUP1 = OPNRUP1(1:nchar) // SR3(1:4)
c     CALL SRTOUT(OPNRUP1)
C
c     OPNRUP1 = OPNRUP1(1:nchar) // tr4(1:4)
c     CALL TRANS(OPNRUP1)
C
c     STOP
c     END
****

```

```
C.....  
C  
C   declarations for DEGADISIN  
C  
C  
C   parameter(      isen= 30,  ! dimension of /sen1/ and /sen2/  
1                  pi= 3.14 159 2654)  
C  
****
```

AD-A171 524

DEVELOPMENT OF AN ATMOSPHERIC DISPERSION MODEL FOR  
HEAVIER-THAN-AIR GAS M. (U) ARKANSAS UNIV FAYETTEVILLE  
DEPT OF CHEMICAL ENGINEERING J A HAVENS ET AL. MAY 85

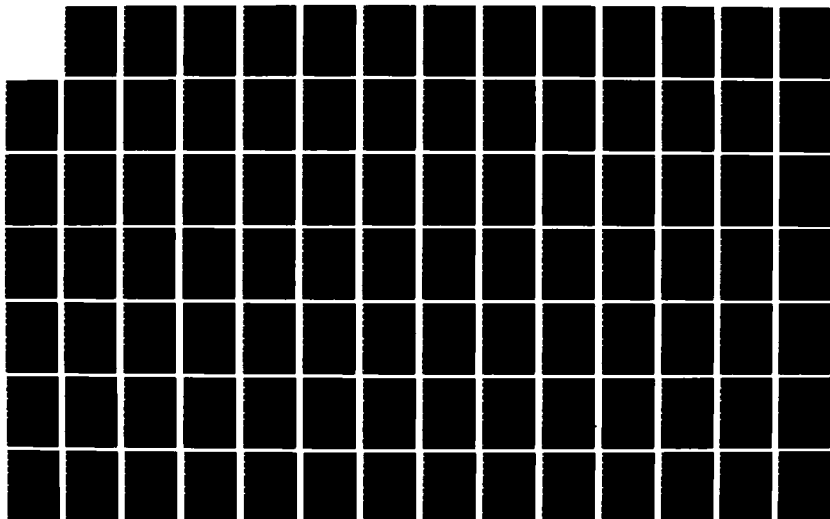
2/3

UNCLASSIFIED

USCG-D-24-85 DTCG23-80-C-20029

F/G 20/4

NL







\*\*\*\*\*  
 \*\*\*\*\*  
 \*\*\*\*\*

C

C PROGRAM DEGADISIN

C

C Program description:

C

C DEGADISIN acts as an interactive input module to the programs  
 C which make up the DEGADIS model. The user is guided through a  
 C series of questions which supply the model with the necessary  
 C input information.

C

C

C Program usage:

C

C Consult Volume III of the Final Report to U. S. Coast Guard  
 C contract DT-CG-23-80-C-20029 entitled 'Development of an  
 C Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.

C

C J. A. Havens

C T. O. Spicer

C

C University of Arkansas

C 227 Engineering Building

C Department of Chemical Engineering

C Fayetteville, AR 72701

C

C April 1985

C

C

C This project was sponsored by the U. S. Coast Guard and the Gas  
 C Research Institute under contract DT-CG-23-80-C-20029.

C

C

C Disclaimer:

C

C This computer code material was prepared by the University of  
 C Arkansas as an account of work sponsored by the U. S. Coast Guard  
 C and the Gas Research Institute. Neither the University of Arkansas,  
 C nor any person acting on its behalf:

C

C a. Makes any warranty or representation, express or implied,  
 C with respect to the accuracy, completeness, or usefulness  
 C of the information contained in this computer code material,  
 C or that the use of any apparatus, method, numerical model,  
 C or process disclosed in this computer code material may not  
 C infringe privately owned rights; or

C

C b. Assumes any liability with respect to the use of, or for  
 C damages resulting from the use of, any information;

C

1 -- sys\$degadis:DEGADISIN.FOR

```

C      apparatus, method, or process disclosed in this computer
C      code material.
C
C
C*****
C*****
C*****
C
C
C
C      INITIAL INPUT FOR DEGADIS ROUTINES
C
C      note: this series of programs relies on the system wide
C            logical symbol SYS$DEGADIS which denotes the source
C            and executable code for these images.
C
C      PROGRAM DEGADISIN
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'SYS$DEGADIS:degadis.in.dec'
C
C      COMMON
C      $/TITL/ TITLE
C      $/GEN1/ ET(2,isn),R1T(2,isn)
C      $/GEN2/ DEN(5,isn)
C      $/ITI/ T1,TINP,TSRC,TOBS,TSRT
C      $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/COM_PROP/ sas_mw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C      $ sas_ufl,sas_lfl,sas_zsp,sas_name
C      $/COM_SS/ ess,slen,suid,outcc,outcz,outb,outl
C      $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      $/COM_six/ six_coeff,six_pow,six_ain_dist,six_flag
C      $/NEND/ POUND,POUND
C
C      character*80 TITLE(4)
C
C      character*3 sas_temp
C      character*4 pound
C      character*24 TSRC,TINP,TOBS,TSRT
C
C      REAL*8 ML,K
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C      c check1
C      c check2=t      cloud type release with no liquid source; SRC1  DEGADIS1
C      c again        local communications in SSSUP                      SSSUP
C      c check3      local communications between SRC1 and NOBL          DEGADIS1
C      c check4=t    steady state simulation                            DEGADISIN
C
C      2 -- sys$degadis:DEGADISIN.FOR

```

```

c check5=1      operator sets sort parameters      ESTR73
c
  data CHECK1/.false./,CHECK2/.false./,AGAIN/.false./
  data CHECK3/.false./,CHECK4/.false./,CHECK5/.false./
c
  character*100 OPNRUP
  character OPNRUP1(100)
  equivalence (opnrup1(1),opnrup)
  character*4 INP,er1,er2,er3,com,scl,sr3,lis
  character*4 dummy
  character*3 plus
  character*2 con
  DATA POUND/'/' '/' ,POUNDN/-1.E-20/
c
  DATA ET/isen%0.,isen%0./,R1T/isen%0.,isen%0./
  data DEN/isen%0.,isen%0.,isen%0.,isen%0.,isen%0./
  DATA INP/''.INP'',er1'',er1'',er2'',er2'',er3'',er3''/
  data scl/''.scl'',sr3'',sr3'',lis'',lis''/
  data com/''.com''/
  data plus/' ' + '' ,con/' -'/
c
C*** GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES
c
  WRITE(6,800)
  WRITE(6,810)
  READ(5,820) NCHAR,OPNRUP
  OPNRUP = OPNRUP(1:NCHAR) // INP(1:4)
c
C*** NOW GET THE REST OF THE DESIRED INFORMATION
c
  CALL IOT(OPNRUP)
  WRITE(6,1000)
  if(check4) then
    write(6,1001) ! continuous
  else
    if(u0 .eq. 0.) then
      write(6,1009)
    else
      WRITE(6,1002) ! transient
    endif
  endif
  write(6,1010)
c
C*** FORMATS
c
  800 FORMAT(///,16X,'Dense GAS DISpersion Model input module.')
  810 FORMAT(//,' Enter the simulation name',
    $' : [DIR]RUNNAME ',%)
  820 FORMAT(Q,A40)
  1000 FORMAT(' ',/,
    $' In addition to the information just obtained,',
  )
3 -- sys$desadis:DEGADISIN.FOR

```

```

$' DEGADIS',/, ' requires a series of numerical parameter',
$' files which use',/, ' the',
$' same name as [DIR]RUNNAME given above. ',/,
$' For convenience, example parameter files are included for',/,
$' each step. They are:')
1001 FORMAT(10X,'EXAMPLE.ER1 and',/,10X,'EXAMPLE.ER2')
1002 format(10X,'EXAMPLE.ER1',/,10X,'EXAMPLE.ER2, and',/,
$10X,'EXAMPLE.ER3')
1009 format(10X,'EXAMPLE.ER1')
1010 format(' Note that each of',
$' these files can be edited during the course of the',/,
$' simulation if a parameter proves to be out of specification.',/)
C
C
      write(6,1200)
1200 format(' Do you want a command file to be generated to execute',
$' the procedure? <Y or n> ',)
      READ(5,1210) dummy
1210 format(a4)
      if(dummy.eq.'n' .or. dummy.eq.'N') goto 3000
      openrup = openrup(1:nchar) // com(1:4)
      write(6,1220) openrup
1220 format(' The command file will be generated under the file',
$' name:',/,10X,a40)
C
      open(unit=8,name=openrup,type='new',
$ carriagecontrol='list',recordtype='variable')
C
      openrup = openrup(1:nchar) // er1(1:4)
C
      write(8,1250) (openrup1(i),i=1,nchar+4)
1250 format('$ copy/log SYS$DEGADIS:example.er1 ',40a1)
      IF(u0 .eq. 0.) then
          write(8,1280)
          write(8,1290) (openrup1(i),i=1,nchar)
          goto 1340
      endif
      openrup = openrup(1:nchar) // er2(1:4)
C
      write(8,1260) (openrup1(i),i=1,nchar+4)
1260 format('$ copy/log SYS$DEGADIS:example.er2 ',40a1)
      openrup = openrup(1:nchar) // er3(1:4)
C
      if(.not.check4) then          ! transient
C
          write(8,1270) (openrup1(i),i=1,nchar+4)
1270      format('$ copy/log SYS$DEGADIS:example.er3 ',40a1)
C
          write(8,1280)
1280      format('$ run SYS$DEGADIS:DEGADIS1')
          write(8,1290) (openrup1(i),i=1,nchar)

```

4 -- sys\$degadis:DEGADISIN.FOR

```

1290      format(40a1)
        write(8,1300)
1300      format('$ run SYS$DEGADIS:DEGADIS2')
        write(8,1290) (opnrup(i),i=1,nchar)
        write(8,1320)
1320      format('$ run SYS$DEGADIS:DEGADIS3')
        write(8,1290) (opnrup(i),i=1,nchar)
c
        opnrup = opnrup(1:nchar) // scl(1:4) //
1      plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
        write(8,1370) (opnrup(i),i=1,2*nchar+13)
1370      format('$ copy/los ',100a1)
c
        opnrup = opnrup(1:nchar) // lis(1:4)
        write(8,1390) (opnrup(i),i=1,nchar+4)
1390      format(' ',40a1)
        else
            write(8,1280)
            write(8,1290) (opnrup(i),i=1,nchar)
c
            write(8,1330)
1330      format('$ run SYS$DEGADIS:SDEGADIS2')
            write(8,1290) (opnrup(i),i=1,nchar)
c
            opnrup = opnrup(1:nchar) // scl(1:4) //
1      plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
            write(8,1370) (opnrup(i),i=1,2*nchar+13)
            opnrup = opnrup(1:nchar) // lis(1:4)
            write(8,1390) (opnrup(i),i=1,nchar+4)
        endif
c
1340 close(unit=8)
        write(6,1350)
1350 format(/,' Do you wish to initiate this procedure? ',
        $' <Y or N> ', $)
        REad(5,1210) dummy
        if(dummy.eq.'Y' .or. dummy.eq.'Y') goto 2000
        goto 3000
2000 opnrup = '8' // opnrup(1:nchar) // ' '
        istat = lib$do_command(opnrup)
        write(6,2100)
2100 format(/,' ?DEGADISIN? command file failed to start.')
c
3000 continue
        CALL EXIT
        END
####

```

```

C.....
C
C ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
C SUBROUTINE ESTRT1(OPNRUP)
C
C Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C include 'sys$desadis:DEGADIS1.dec'
C
C parameter(      iend= 22,
1                iend1= iend+1,
1                iiiiend= 2,
1                iiiiend1= iiiiend+1,
1                iiiiend= 2,
1                iiiiend1= iiiiend+1,
1                iend= 2,
1                iend1= iend+1,
1                jend= 4,
1                jend1= jend+1,
1                mend= 5,
1                mend1= mend+1)
C
C BLOCK COMMON
C
C COMMON
$/ERROR/STPIN,ERBND,STPHX,WTRG,WTta,WTya,WTyc,WTeb,WTmb,WTuh,XLI,
$ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
$ htcut,ERNOBL,NOBLpt,crfser,epsilon
$/vucom/ vua,vub,vuc,vud,vudelta,vufas
$/szfc/ szst0,szerr,szstmax,szsz0
$/alphcom/ ialpfl,alpco
$/phicom/ iphifl,dellay
$/srd_con/ ce, delrhomin
C
C EQUIVALENCE
$(RLBUF(1),STPIN), !MAIN - RKGST - INITIAL STEP SIZE
$(RLBUF(2),ERBND), !MAIN - RKGST - ERROR BOUND
$(RLBUF(3),STPHX), !MAIN - RKGST - MAXIMUM STEP SIZE
$(RLBUF(4),WTRG), !MAIN - RKGST - WEIGHT FOR RG
$(RLBUF(5),WTta), !MAIN - RKGST - WEIGHT FOR Total mass
$(RLBUF(6),WTya), !MAIN - RKGST - WEIGHT FOR Ya
$(RLBUF(7),WTyc), !MAIN - RKGST - WEIGHT FOR Yc
$(RLBUF(8),WTeb), !MAIN - RKGST - WEIGHT FOR Energy Balance
$(RLBUF(9),WTmb), !MAIN - RKGST - WEIGHT FOR Momentum Balance
$(RLBUF(10),WTuh), !MAIN - RKGST - WEIGHT FOR Ueff*ileff
$(RLBUF(11),XLI), !ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
$(RLBUF(12),XRI), !ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
$(RLBUF(13),EPS) !ALPH - ERROR BOUND USED BY 'RTHI'
equivalence
$(RLBUF(14),ZLOW), !ALPHI - BOTTOM HEIGHT FOR FIT OF ALPHA
$(RLBUF(15),STPINZ), !ALPHI - INITIAL RKGST STEP <0.

1 -- sys$desadis:ESTRT1.FOR

```

```

$(RLBUF(16),ERBNDZ), !ALPHI - ERROR BOUND FOR RKGST
$(RLBUF(17),STPMXZ), !ALPHI - MAXIMUM STEP FOR RKGST
$(RLBUF(18),SRCOER), !SRC10 - OUTPUT error criteria
$(RLBUF(19),SRCSS), !SRC10 - min time for Steady;4*STPMX
$(RLBUF(20),SRCcut), !SRC10 - min height for blanket
$(RLBUF(21),htcut), !SRC1 - min height for blanket heat transfer
$(RLBUF(iend),ERNODL), !NOBL - CONVERGENCE CRITERIA
$(RLBUF(i1),crfser), !CRFG - Error criteria for building tables
$(RLBUF(iiiiend),epsilon)!SRC1 - Coefficient in Air entrainment

C
    equivalence
$(rlbufa(1),ce), !SRC1 - Coefficient gravity slumping EQ
$(RLBUFA(iiiend),delrhomin) ! stop spread for delrho<delrhomin

C
    equivalence
$(rlbuf1(1),szst0), ! SZF - Initial step size
$(rlbuf1(2),szerr), ! SZF - Error criteria
$(rlbuf1(3),szstmax), ! SZF - Maximum step size
$(rlbuf1(4),szsz0) ! SZF - Initial value of rho*delay*UHeff

C
C
    equivalence
$(rlbuf4(1),vua), ! Constant Av in SRC1
$(rlbuf4(2),vub), ! Constant Bv in SRC1
$(rlbuf4(3),vuc), ! Constant Ev in SRC1
$(rlbuf4(4),vud), ! Constant Dv in SRC1
$(rlbuf4(5),vudelta) ! Constant DELTA v in SRC1

C
    character*40 OPNRUP
    character DUMMY(1:132)
    DIMENSION RLBUF(iend), rlbufi(iiiiend), rlbufa(iiiend)
    dimension rlbuf1(jend)
    dimension rlbuf4(mend)

C
    logical vuflag

C
    OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD',err=2000)

C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
    I = 1
100 CONTINUE
    READ(9,1000,END=350) NCHAR,DUMMY
    IF(DUMMY(1) .EQ. '!') GO TO 100
    DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
    I = I + 1
    if(i .eq. iend1) goto 110
    GO TO 100

110 CONTINUE
    READ(9,1000,END=350) NCHAR,DUMMY

2 -- sys$desadis:ESTRT1.FOR

```

```

IF(DUMMY(1) .EQ. '!') GO TO 110
DECODE(20,1010,DUMMY,ERR=400) ptnobl
NOBLPT = INT(PTNOBL)

```

```

I = 1
120 CONTINUE
READ(9,1000,END=350) NCHAR,DUMMY
IF(DUMMY(1) .EQ. '!') GO TO 120
DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
I = I + 1
if(i .eq. iend1) goto 140
GO TO 120

```

```

C
C*** READ A LINE AND DETERMINE ITS PURPOSE for /sprd_con/
C

```

```

140 I = 1
150 CONTINUE
READ(9,1000,END=350) NCHAR,DUMMY
IF(DUMMY(1) .EQ. '!') GO TO 150
DECODE(20,1010,DUMMY,ERR=400) RLBUF2(I)
I = I + 1
if(i .eq. iend1) goto 190
GO TO 150

```

```

C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill szfc
C

```

```

190 I = 1
200 CONTINUE
READ(9,1000,END=350) NCHAR,DUMMY
IF(DUMMY(1) .EQ. '!') GO TO 200
DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
I = I + 1
if(i .eq. jend1) goto 230
GO TO 200

```

```

C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /alphcom/
C

```

```

230 I = 1
240 CONTINUE
READ(9,1000,END=350) NCHAR,DUMMY
IF(DUMMY(1) .EQ. '!') GO TO 240
DECODE(20,1010,DUMMY,ERR=400) Ralpfl

ialpfl = int(ralpfl)

```

```

250 CONTINUE
READ(9,1000,END=350) NCHAR,DUMMY
IF(DUMMY(1) .EQ. '!') GO TO 250
DECODE(20,1010,DUMMY,ERR=400) alpc

```

```

C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /phicom/

```

```

3 -- sys$desadis:ESTRT1.FOR

```



C

```

260 I = 1
270 CONTINUE
  READ(9,1000,END=350) NCHAR,DUMMY
  IF(DUMMY(1) .EQ. '!') GO TO 270
  DECODE(20,1010,DUMMY,ERR=400) Rphifl

  i:phifl = int(r:phifl)

```

```

275 CONTINUE
  READ(9,1000,END=350) NCHAR,DUMMY
  IF(DUMMY(1) .EQ. '!') GO TO 275
  DECODE(20,1010,DUMMY,ERR=400) dellay

```

C

C\*\*\* READ A LINE AND DETERMINE ITS PURPOSE to fill /vucom/

C

```

280 I = 1
290 CONTINUE
  READ(9,1000,END=350) NCHAR,DUMMY
  IF(DUMMY(1) .EQ. '!') GO TO 290
  DECODE(20,1010,DUMMY,ERR=400) RLBUF4(I)
  I = I + 1
  if(i .eq. send1) goto 300
  GO TO 290

```

C

C\*\*\* EXIT THE PROCEEDINGS

C

```

300 CONTINUE
  CLOSE(UNIT=9)
  RETURN

```

C

```

350 call trap(20)

```

C

```

400 CONTINUE
  CALL trap(21)

```

C

```

1000 FORMAT(Q,132A1)
1010 FORMAT(10X,G10.4)

```

C

```

2000 call trap(22)
  END

```

\*\*\*\*

```

C.....
C
C ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
C SUBROUTINE ESTRT2(OPNRUP)
C
C Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C include 'sys$desadis:DEGADIS2.dec/list'
C
C parameter (      ienda= 18,
C 1              ienda1= ienda+1,
C 2              iendb= 7,
C 3              iendb1= iendb+1)
C
C common
C $/ERROR/SYOER,ERRQ,SZOER,WTAIQ,WTQOO,WT SZO,ERRP,SMXP,
C $ WTSZP,WTSYP,WTBEP,WT DH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WDHG
C $/STP/STPQ,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C $/CNOBS/NOBS
C
C EQUIVALENCE
C $(RLBUF(1),SYOER), !SSSUP - RKGST - INITIAL SY
C $(RLBUF(2),ERRQ), !SSSUP - RKGST(OBS) - ERROR BOUND
C $(RLBUF(3),SZOER), !SSSUP - RKGST(OBS) - INITIAL SZ
C $(RLBUF(4),WTAIQ), !SSSUP - RKGST(OBS) - WEIGHT FOR AI
C $(RLBUF(5),WTQOO), !SSSUP - RKGST(OBS) - WEIGHT FOR Q
C $(RLBUF(6),WT SZO), !SSSUP - RKGST(OBS) - WEIGHT FOR SZ
C $(RLBUF(7),ERRP), !SSSUP - RKGST(PSS) - ERROR BOUND
C $(RLBUF(8),SMXP), !SSSUP - RKGST(PSS) - MAXIMUM STEP
C $(RLBUF(9),WTSZP), !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
C $(RLBUF(10),WTSYP), !SSSUP - RKGST(PSS) - WEIGHT FOR SY
C $(RLBUF(11),WTBEP), !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
C $(RLBUF(12),WT DH), !SSSUP - RKGST(PSS) - WEIGHT FOR DH
C $(RLBUF(13),ERRG), !SSSUP - RKGST(SSG) - ERROR BOUND
C $(RLBUF(14),SMXG), !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
C $(RLBUF(15),ERTDNF), !TDNF - CONVERGENCE CRITERIA
C $(RLBUF(16),ERTUPF), !TUPF - CONVERGENCE CRITERIA
C $(RLBUF(17),WTRUH), !SSSUP - RKGST(SSG) - WEIGHT FOR RUH
C $(RLBUF(ienda),WDHG)!SSSUP - RKGST(SSG) - WEIGHT FOR DH
C
C EQUIVALENCE
C $(RLBUF(1),STPQ), !SSSUP - RKGST(OBS) - INITIAL STEP
C $(RLBUF(2),STPP), !SSSUP - RKGST(PSS) - INITIAL STEP
C $(RLBUF(3),ODLP), !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
C $(RLBUF(4),ODLLP), !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
C $(RLBUF(5),STPG), !SSSUP - RKGST(SSG) - INITIAL STEP
C $(RLBUF(6),ODLG), !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
C $(RLBUF(iendb),ODLLG)!SSSUP - RKGST(SSG) - MAXIMUM DISTANCE TO OUT
C
C 1 -- sys$desadis:ESTRT2.FOR

```

```

character*40 OPNRUP
character dummy(1:132)
DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
      I = 1
200 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
      I = I + 1
      IF(I .EQ. ienda1) GO TO 200
      GO TO 100
C
C*** NOW, FILL RLBUF1
C
200 I = 1
210 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 210
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      IF(I .EQ. iendb1) GO TO 260
      GO TO 210
C
C*** NOW, PICK UP NOBS
C
260 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 260
      DECODE(20,1010,DUMMY,ERR=400) RBUF
      NOBS = INT(RBUF)
C
C*** EXIT THE PROCEEDINGS
C
      CLOSE(UNIT=9)
      RETURN
C
300 call trap(20)
400 CALL trap(21)
C
1000 FORMAT(Q,132A1)
1010 FORMAT(10X,G10.4)
      END
****

```

```

C.....
C
C   ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
C   SUBROUTINE ESTR2SS(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
C   include 'sys$desadis:DEGadis2.dec/list'
C
C   parameter(      ienda= 18,
C   1              ienda1= ienda+1,
C   2              iendb= 7,
C   3              iendb1= iendb+1)
C
C   COMMON
C   $/ERROR/SYOEER,ERRP,SMXP,WTSPZ,WTSPY,WTBEP,WDH,ERRG,SMXG,
C   $ WTRUH,WDHG
C   $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C
C   character*40 OPNRUP
C   character DUMMY(1:132)
C   DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
C   OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
      I = 1
100 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
      I = I + 1
      IF(I.EQ. ienda1) GOT0200
      GO TO 100
C
C*** NOW, FILL RLBUF1
C
200 i = 1
210 READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 210
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      if(i.eq. iendb1) goto 300
      GO TO 210
C
1 -- sys$desadis:ESTRT2SS.FOR

```

\*\*\* EXIT THE PROCEEDINGS

C

300 CONTINUE

```

sy0er = rdbuf(1) !SSSUP - RKGST - INITIAL SY
errp = rdbuf(7) !SSSUP - RKGST(PSS) - ERROR BOUND
saxp = rdbuf(8) !SSSUP - RKGST(PSS) - MAXIMUM STEP
wtSZP = rdbuf(9) !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
wtSVP = rdbuf(10) !SSSUP - RKGST(PSS) - WEIGHT FOR SY
wtBEP = rdbuf(11) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
wtDH = rdbuf(12) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
errS = rdbuf(13) !SSSUP - RKGST(SSG) - ERROR BOUND
saxS = rdbuf(14) !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
wtRUH = rdbuf(17) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
wtDHG = rdbuf(18) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF

```

C

```

step = rdbuf1(2) !SSSUP - RKGST(PSS) - INITIAL STEP
odlp = rdbuf1(3) !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
odllp = rdbuf1(4) !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
stepS = rdbuf1(5) !SSSUP - RKGST(SSG) - INITIAL STEP
odls = rdbuf1(6) !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
odlls = rdbuf1(7) !SSSUP - RKGST(SSG) - MAXIMUM DISTANCE TO OUT

```

C

```

CLOSE(UNIT=9)
RETURN

```

C

350 call trap(20) ! premature EOF

C

400 CALL trap(21)

C

```

1000 FORMAT(Q,132A1)
1010 FORMAT(10X,G10.4)

```

C

END

\*\*\*\*

```

C.....
C
C   ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
C   SUBROUTINE ESTR3(OPNRUP)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       COMMON
C       $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C       $/com_six/ six_coeff,six_pow,six_min_dist,six_flag
C       $/ERROR/ERT1,ERDT,ERNTIM
C
C       EQUIVALENCE
C       $(RLBUF(1),ERT1),      !FIRST SORT TIME - USER OPTION
C       $(RLBUF(2),ERDT),      !SORT TIME DELTA - USER OPTION
C       $(RLBUF(3),ERNTIM)     !NUMBER OF SORT TIMES - USER OPTION
C
C       LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C       character DUMMY(1:132)
C       character*40 opnrup
C       DIMENSION RLBUF(3),RBUF(6)
C
C       OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C       I = 1
C       100 CONTINUE
C       READ(9,1000,END=300) NCHAR,DUMMY
C       IF(DUMMY(1).EQ. '!') GO TO 100
C       DECODE(20,1010,DUMMY,ERR=400) RBUF(I)
C       I = I + 1
C       GO TO 100
C
C*** EXIT THE PROCEEDINGS AND DETERMINE CHECKS
C       300 CONTINUE
C
C       DO 310 I = 1,3
C       310 RLBUF(I) = RBUF(I)
C       CHECK5 = .FALSE.                ! IN ORDER FOR FLAG TO WORK
C       IF(RBUF(4).EQ. 1.) CHECK5 = .TRUE.
C
C       six_flag = rbuf(5)
C       CLOSE(UNIT=9)
C       RETURN
C
C       400 CALL trap(21)
C       1000 FORMAT(0,132A1)
C       1010 FORMAT(10X,G10.4)
C       END
C
C****
C
C 1 -- sys$des$adis:ESTRT3.FOR

```

```

C
C .....
C
C SUBROUTINE GAMMA
C
C .....
C
C This routine was originally supplied by Digital Equipment
C Corporation as part of the Scientific Subroutine Package
C available for RT-11 as part of the Fortran Enhancement
C Package. It was upgraded for use in this package.
C
C .....
C
C PURPOSE
C     COMPUTES THE GAMMA FUNCTION FOR A GIVEN ARGUMENT
C
C USAGE
C     GF = GAMMA(XX)
C
C DESCRIPTION OF PARAMETERS
C     XX -THE ARGUMENT FOR THE GAMMA FUNCTION
C
C IER-RESULTANT ERROR CODE WHERE
C     IER=0 NO ERROR
C     IER=1 XX IS WITHIN .000001 OF BEING A NEGATIVE INTEGER
C     IER=2 XX GT 34.5, OVERFLOW
C     IF IER .NE. 0 PROGRAM TAKES A DIP IN THE POOL!
C
C REMARKS
C     NONE
C
C SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C     NONE
C
C METHOD
C     THE RECURSION RELATION AND POLYNOMIAL APPROXIMATION
C     BY C.HASTINGS,JR., 'APPROXIMATIONS FOR DIGITAL COMPUTERS',
C     PRINCETON UNIVERSITY PRESS, 1955
C
C MODIFIED TO FUNCTION FORM FROM ORIGINAL SUBROUTINE FORM
C
C .....
C
C FUNCTION GAMMA(XX)
C
C     Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C     IF(XX-34.5)6,6,4
C 4 IER=2
C     GAMMA=1.E38
C
C 1 -- sys$desadis:GAMMA.FOR

```

```

      GO TO 1000
6  X=XX
   ERR=1.0E-6
   IER=0
   GAMMA=1.0
   IF(X-2.0)50,50,15
10 IF(X-2.0)110,110,15
15 X=X-1.0
   GAMMA=GAMMA*X
   GO TO 10
50 IF(X-1.0)60,120,110
C
C      SEE IF X IS NEAR NEGATIVE INTEGER OR ZERO
C
60 IF(X-ERR)62,62,80
62 Y=FLOAT(INT(X))-X
   IF(ABS(Y)-ERR)130,130,64
64 IF(1.0-Y-ERR)130,130,70
C
C      X NOT NEAR A NEGATIVE INTEGER OR ZERO
C
70 IF(X-1.0)80,80,110
80 GAMMA=GAMMA/X
   X=X+1.0
   GO TO 70
110 Y=X-1.0
   GY=1.0+Y*(-0.5771017+Y*(+0.9858540+Y*(-0.8764218+Y*
   *(+0.8328212+Y*(-0.5684729+Y*(+0.2548205+Y*(-0.05149930))))))
   GAMMA=GAMMA*GY
120 RETURN
130 IER=1
1000 CONTINUE
   IF(IER.EQ.1) WRITE(5,1100)
   IF(IER.EQ.2) WRITE(5,1110)
1100 FORMAT(5X,'?GAMMA?--ARGUMENT LESS THAN 0.000001')
1110 FORMAT(5X,'?GAMMA?--ARGUMENT GREATER THAN 34.5--OVERFLOW')
   CALL EXIT
   END
****

```



```

C.....
C
C   SUBROUTINE TO ESTABLISH THE TIME SORT PARAMETERS
C
C   SUBROUTINE GETTIM
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sys$desadis:DEGADIS3.dec/list'
C
C   COMMON
C   %/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
C   %/SORTIN/ TIM(maxnt),NTIM,ISTR1
C   %/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C   %/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   %/ERROR/ ERT1,ERDT,ERNTIM
C   %/ALP/ ALPHA,alpha1
C   %/CNOBS/ NOBS
C
C   LOGICAL CHECK1,CHECK2,CHECK3,CHECK4,CHECK5,AGAIN
C
C   DATA T1/0./,DT/0./,TF/0./
C
C*** IF CHECK5 IS SET, GET THE TIME SORT PARAMETERS FROM /ERROR/
C
C   IF(.NOT.CHECK5) GO TO 90
C
C   T1 = ERT1
C   DT = ERDT
C   NTIM = INT(ERNTIM)
C   GO TO 95
C
C
C
C*** This subroutine sets the default time sort windows.
C
C*** The first sort time is set for potential low wind speed cases,
C*** while the last sort time is set for potential high wind speed
C*** cases. The first sort time is taken to be when the first
C*** observer passes through x=RMAX. The last sort time is taken
C*** to be when the last observer passes through x=6*RMAX.
C*** The default value for the number of sort times is set to 10.
C*** Obviously, these values generate some sort times which will be
C*** useless; hopefully, these values will show the user where to
C*** look on the next go around.
C
C   90 CONTINUE
C
C   T1 = T0(1) + (2.*RMAX)**(1./ALPHA1)/ALEPH
C   TF = T0(NOBS) + (6.*RMAX)**(1./ALPHA1)/ALEPH
C   NTIM = 10
C
1 -- sys$desadis:GETTIM.FOR

```

C-47

```
C      DT = (TF-T1)/FLOAT(NTIM-1)
      DT = FLOAT(INT(DT+.5))
C
      IF(DT .GE. 5.) GO TO 95
      DT = 5.
      NTIM = INT((TF - T1)/DT) + 1
C
95  CONTINUE
C
      T1 = FLOAT(INT(T1))      !MAKE T1 AN INTEGER VALUE
C
      DO 100 I = 1,NTIM
      TIM(I) = DT*FLOAT(I-1) + T1
100 CONTINUE
C
      RETURN
      END
****
```

SUBROUTINE HEAD( mass0 )

Implicit Real\*8 ( A-H, O-Z ), Integer\*4 ( I-N )

```

C      include 'sys$desadis:DEGADIS1.dec'
C      include '($sdef)'

C      COMMON
C      $/GEN3/ RADG(2,max1),QSTR(2,max1),srcden(2,max1),srcwc(2,max1),
C      $ srcwa(2,max1),srcenth(2,max1)
C      $/TITL/ TITLE
C      $/GEN1/ ET(2,isen),R1T(2,isen)
C      $/GEN2/ DEN(5,isen)
C      $/ITI/ T1,TINP,TSRC,TOBS,TSRT
C      $/ERROR/ STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtac,uteb,utab,utuh,XLI,
C      $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srccs,srccut,
C      $ htcut,ERNOBL,NOBLt,crfser,epsilon
C      $/PARM/ UO,ZO,ZR,HL,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/com_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpp,
C      $ sas_ufl,sas_lfl,sas_zsp,sas_name
C      $/comata/ istab,tamb,pamb,humid,isofl,tsurf,htf1,htco,iutfl,wtco
C      $/com_ss/ ess,slen,suid,outcc,outcz,outb,outl,swcl,swal,senl,srhl
C      $/phas/ check1,check2,asain,check3,check4,check5
C      $/NEND/ POUNDN,POUND
C      $/ALP/ ALPHA,alpha1
C      $/alphcom/ ialpfl,alpco
C      $/phicom/ iphifl,dellay
C      $/srd_con/ ce, delrhomin

C      character*80 TITLE(4)

C      character*4 pound
C      character*24 TINP,TSRC,TOBS,TSRT
C      character*3 sas_name
C      character*1 stabil(6)
C      character*24 id

C      logical check1,check2,asain,check3,check4,check5

C      REAL*8 K,HL

C      data stabil/'A','B','C','D','E','F'/
C      data iparm/0/

C      if(iparm .eq. 1) goto 190
C      WRITE(8,1100)
1100 FORMAT(1H0,'*****',9X,'U O A _ D E G A D I S ',
C      $2X,'M O D E L ',2X,'O U T P U T ',2X,'- - ',2X,'V E R S I O N ',
C      $2X,'1.2',8X,'*****')

C
1 -- sys$desadis:HEAD.FOR

```

```

C      WRITE(8,1111)
C
C      WRITE(8,1102) tsrc
1102 FORMAT(1H , '*****', 23X,
$ '*****', 1X, a24, 1X,
$ '*****', 23X, '*****')
C
C      WRITE(8,1111)
C
C      WRITE(8,1112) TINP
C      WRITE(8,1114) TSRC
C      IF(tOBS(1:2).NE. ' ' .and. .not.check4) WRITE(8,1116) TOBS
C      IF(tOBS(1:2).NE. ' ' .and. check4) WRITE(8,1117) TOBS
C      IF(tSRT(1:2) .NE. ' ') WRITE(8,1118) TSRT
1112 FORMAT(1H , 'Data input on', 22X, a24)
1114 FORMAT(1H , 'Source program run on', 14X, a24)
1116 FORMAT(1H , 'Pseudo Steady-State program run on ', a24)
1117 FORMAT(1H , 'Steady-State program run on ', 7X, a24)
1118 FORMAT(1H , 'Time sort program run on ', 11X, a24)
C      WRITE(8,1111)
C
C      write(8,1119)
1119 format(//,
11h ,10x,22('****'),/,
21h ,10x,'*',t121,'*',/,
31h ,10x,'*',t20,'NOTE:',t121,'*',/,
21h ,10x,'*',t20,'-----',t121,'*',/,
21h ,10x,'*',t121,'*',/,
21h ,10x,'*',t20,'>',t25,'All Calculations are lim',
3'ted to circular liquid sources.',t121,'*',/,
21h ,10x,'*',t121,'*',/,
11h ,10x,22('****'),//)
C      WRITE(8,1110)
C      WRITE(8,1111)
C
C      1110 FORMAT(1H0,10X,'TITLE BLOCK')
C      1111 FORMAT(1H )
C
C      DO 100 I = 1,4
C      WRITE(8,1120) TITLE(I)
C      100 CONTINUE
C
C      1120 FORMAT(1H ,A80)
C
C      WRITE(8,1111)
C      WRITE(8,1130) UO
C      WRITE(8,1140) Z0
C      WRITE(8,1150) ZR
C      write(8,1155) stabil(istab)
C      WRITE(8,1160) ML

```

```

WRITE(8,1170) DELTA
WRITE(8,1180) BETA
WRITE(8,1190) ALPHA
WRITE(8,1192) USTAR
WRITE(8,1194) tamb
if(isofl.eq.0 .and. ihtfl.ne.0) write(8,1195) tsurf
WRITE(8,1196) pamb
WRITE(8,1198) humid
vapor = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
relhumid = 100.* humid/(0.622*vapor / (pamb- vapor))
write(8,1199) relhumid

```

C

```

1130 FORMAT(1H ,5X,'Wind velocity at reference height ',20X,F6.2,2X,
      $'m/s')
1140 FORMAT(1H ,5X,'Reference height ',37X,F6.2,2X,'m')
1150 FORMAT(1H0,5X,'Surface roughness length ',25X,1P610.3,2X,'m')
1155 FORMAT(1H0,5X,'Pasquill Stability class ',25X,4X,a1)
1160 FORMAT(1H0,5X,'Monin-Obukhov length ',29X,1P610.3,2X,'m')
1170 FORMAT(1H ,5X,'Gaussian distribution constants ',4X,'Delta',
      $10X,F9.5,2X,'m')
1180 FORMAT(1H ,5X,32X,4X,'Beta',11X,F9.5)
1190 FORMAT(1H0,5X,'Wind velocity power law constant',4X,'Alpha',
      $10X,F9.5)
1192 FORMAT(1H ,5X,'Friction velocity',15X,4X,5X,10X,F9.5,2X,'m/s')
1194 FORMAT(1H0,5X,'Ambient Temperature ',35X,F6.2,2X,'K')
1195 FORMAT(1H0,5X,'Surface Temperature ',35X,F6.2,2X,'K')
1196 FORMAT(1H ,5X,'Ambient Pressure ',37X,F6.3,2X,'atm')
1198 FORMAT(1H ,5X,'Ambient Absolute Humidity',25X,1P610.3,2X,
      $'kg/kg BDA')
1199 FORMAT(1H ,5X,'Ambient Relative Humidity',25X,4X,F6.2,2X,'%')

```

C

```

WRITE(8,1111)

```

C

```

if(isofl .eq. 0) goto 135
WRITE(8,1200)
WRITE(8,1205)
ii = -1
DO 130 I = 1,isen
  IF(DEN(1,I).st. 1.) goto 148
  ii = ii+1
  if(ii.eq. 3) then
    write(8,1211)
    ii = 0
  endif
130 WRITE(8,1210) DEN(1,I),DEN(2,I),den(3,i)
  goto 148
135 write(8,1207)
  write(8,1208)
  ii = -1
  DO 138 I = 1,isen
    IF(DEN(1,I).st. 1.) goto 148

```

```

      ii = ii+1
      if(ii.eq. 3) then
        write(8,1211)
        ii = 0
      endif
138 WRITE(8,1212) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
148 continue
C
1200 FORMAT(1H ,5X,'Input:      ',6X,3X,'Mole fraction',4X,
1      'CONCENTRATION OF C',6X,'GAS DENSITY')
1205 FORMAT(1H ,14X,20X,2(13X,'ks/m**3'))
1207 FORMAT(1H ,5X,'Adiabatic Mixins:',3X,'Mole fraction',3X,
1      'CONCENTRATION OF C',6X,'GAS DENSITY',5X,
1      6X,'Enthalpy',6X,1X,'Temperature')
1208 FORMAT(1H ,14X,20X,2(13X,'ks/m**3'),7X,8X,'J/ks',8X,9X,'K')
1210 FORMAT(1H ,14X,3(12X,F8.5))
1211 format(1H )
1212 FORMAT(1H ,14X,3(12X,F8.5),6X,3X,1P13.5,7X,1P13.5)
C
      WRITE(8,1111)
      WRITE(8,1220) smass0
      WRITE(8,1230)
C
      DO 150 I=1,IGEN
      IF(R1T(1,I).EQ.POUNDN .AND. R1T(2,I).EQ.POUNDN) GO TO 160
150 WRITE(8,1240) ET(1,I),ET(2,I),R1T(2,I)
160 CONTINUE
C
1220 FORMAT(1H , 'Source input data points',//,
1      1h ,15X,'Initial mass in cloud: ',1P13.5,//,
1      1h ,24X,8X,'TIME',10X,'SOURCE S',
2      'TRENGTH',6X,'SOURCE RADIUS')
1230 FORMAT(1H ,34X,'s',17X,'ks/s',18X,'m')
1240 FORMAT(1H ,24X,3(4X,1P12.5,4X))
1241 format(1h0,5X,'Calculation procedure for ALPHA: ',I2)
1242 format(1h0,5X,'Entrainment prescription for PHI: ',I2)
1244 format(1h0,5X,'Layer thickness ratio used for average depth: ',
1      1P13.5)
1250 format(1h0,5X,'Air entrainment coefficient used: ',f5.3)
1251 format(1h0,5X,'NON Isothermal calculation')
1252 format(1h0,5X,'Gravity slumping velocity coefficient used: ',f5.3)
1253 format(1h0,5X,'Heat transfer calculated with fixed coefficient: ',
1      1P13.5,' J/m**2/s/K')
1254 format(1h0,5X,'Heat transfer not included')
1255 format(1h0,5X,'Heat transfer calculated with correlation: ',I2)
1256 format(1h0,5X,'Isothermal calculation')
1257 format(1h0,5X,'Water transfer calculated with fixed coefficient: ',
1      1P13.5,' /m**2/s/atm')
1258 format(1h0,5X,'Water transfer not included')
1259 format(1h0,5X,'Water transfer calculated with correlation')
C
4 -- sys$desadis:HEAD.FOR

```

```

WRITE(8,1111)
write(8,1241) ialpfl
write(8,1242) iphifl
write(8,1244) dellay
write(8,1250) epsilon
write(8,1252) ce
if(isofl.eq. 0) write(8,1251)
if(isofl.ne. 0) write(8,1256)
if(ihtfl.lt. 0) write(8,1253) htco
if(ihtfl.eq. 0) write(8,1254)
if(ihtfl.gt. 0) write(8,1255) ihtfl
if(iwtfl.lt. 0) write(8,1257) wtco
if(iwtfl.eq. 0) write(8,1258)
if(iwtfl.gt. 0) write(8,1259)
WRITE(8,1111)
ipara =1
return

```

```

C
190 continue
if(.not.check4) return
RAD = SQRT(SLEN*SWID/pi)
WRITE(8,1300) ESS,RAD
WRITE(8,1320) SLEN,SWID
astar = ess/outl**2
WRITE(8,1340) OUTCc,OUTSZ,astar
write(8,1350) sucl,sual,senl,srhl
WRITE(8,1360) OUTL,OUTB

```

C  
C  
C

```

1300 FORMAT(1H0,'Source strenth [ks/s] : ',18X,1PG13.5,T60,
$'Equivalent Primary source radius [m] : ',1PG13.5)
1320 FORMAT(1H , 'Equivalent Primary source length [m] : ',4X,
$1PG13.5,T60,'Equivalent Primary source width [m] : ',1X,1PG13.5)
1340 FORMAT(/,' Secondary source concentration [kg/m**3] : ',
$1PG13.5,T60,'Secondary source SZ [m] : ',13X,1PG13.5,//,
1 ' Contaminant flux rate: ',1PG13.5,/)
1350 format(/,' Secondary source mass fractions... contaminant: ',
1 1PG13.6,2X,' air: ',1PG13.5,/, ' ',10X,' Enthalpy: ',
1 1PG13.5,5X,' Density: ',1PG13.5)
1360 FORMAT(1H , 'Secondary source length [m] : ',13X,1PG13.5,T60,
$'Secondary source half-width [m] : ',5X,1PG13.5)

```

C  
C

```

RETURN
END

```

\*\*\*\*

```

C.....
C
C      INPUT SUBROUTINE FOR DEGADIS MODEL
C
C      SUBROUTINE IO(tend,sass0,OPNRUP)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'sys$desadis:DEGADIS1.dec'
C
C      BLOCK COMMON
C
C      COMMON
C      $/TITL/TITLE
C      $/GEN1/ ET(2,isn),R1T(2,isn)
C      $/GEN2/ DEN(5,isn)
C      $/ITI/ T1,TINP,TSRC,TOBS,TSRT
C      $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/COM_PROP/ sas_w,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C      $ sas_ufl,sas_lfl,sas_zsp,sas_name
C      $/COMATA/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wteo
C      $/COM_SS/ ess,slen,swid,outcc,outcz,outb,outl
C      $/PHAS/check1,check2,asain,check3,check4,check5
C      $/COM_SIX/ six_coeff,six_pow,six_min_dist,six_flg
C      $/NEND/POUNDN,POUND
C
C
C      character*80 TITLE(4)
C      character*4 pound
C      character*24 TSRC,TINP,TOBS,TSRT
C      character*3 sas_name
C
C      REAL*8 ML,K
C      logical check1,check2,asain,check3,check4,check5
C
C      character*40 OPNRUP
C
C      OPEN(UNIT=9, NAME=OPNRUP, TYPE='OLD')
C
C      DO 90 I=1,4
C      READ(9,2000) TITLE(I)
C      90 CONTINUE
C      2000 FORMAT(A80)
C
C      READ(9,*) U0,Z0,zr
C      read(9,*) istab
C      READ(9,*) DELTA,BETA,ml
C      read(9,*) six_coeff,six_pow,six_min_dist
C      read(9,*) tamb,pamb,humid
C      read(9,*) isofl,tsurf
C      read(9,*) ihtfl,htco
C
C      1 -- sys$desadis:IO.FOR

```



```

      read(9,*) iwtfl,wlco
      read(9,2020) sas_name
2020 FORMAT(A3)
      read(9,*) sas_aw,sas_temp,sas_rhoe
      read(9,*) sas_cpk,sas_cpp
      read(9,*) sas_ufl,sas_lfl,sas_zsp
C
      if(isofl .eq. 0) then
         rhoe = sas_rhoe * pamb
         rhoa = pamb*(1.+humid)/(.00283+ 0.00456*humid)/tamb
         soto 105
         endif
      READ(9,*) NP
      DO 100 I=1,NP
100  READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,i),den(5,i)
      RHOE = DEN(3,NP)
      RHOA = DEN(3,1)
      den(1,np+1) = 2.
C
105  READ(9,*) CcLOW
C
      read(9,*) smass0
      READ(9,*) NP
      DO 110 I=1,NP
110  READ(9,*) ET(1,I),ET(2,I),R1T(2,I)
      TEND = ET(1,NP-2)
      I = NP + 1
      ET(1,I) = POUNDN
      ET(2,I) = POUNDN
      R1T(1,I) = POUNDN
      R1T(2,I) = POUNDN
C
      DO 120 I=1,NP
120  R1T(1,I) = ET(1,I)
C
      read(9,*) check1,check2,asain,check3,check4,check5
C
      tobs = ' '
      tsrt = ' '
      READ(9,2010) TINP
2010 format(a24)
C
      if(check4) read(9,*) ess,slen,swid
C
      CLOSE(UNIT=9)
      RETURN
      END
####

```

```

SUBROUTINE IOT(OPNRUP)

  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
  include 'sys$desadis:DEGADISIN.dec'
C
  COMMON
  $/TITL/ TITLE
  $/GEN1/ ET(2,isn),R1T(2,isn)
  $/GEN2/ DEN(5,isn)
  $/ITI/ T1,TINP,TSRC,TOBS,TSRT
  $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
  $/COM_PROP/ djs_mw,das_temp,das_rhoe,das_cpk,das_cpp
  $ das_ufl,das_lfl,das_zsp,das_name
  $/COM_SS/ ess,slen,swid,outcc,outsz,outb,outl
  $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
  $/COM_six/ six_coeff,six_pow,six_min_dist,six_flg
  $/NEND/ POUNDN,POUND

C
  character*80 TITLE(4)
  character*3 das_name
  character*4 pound
  character*24 TSRC,TINP,TOBS,TSRT

C
  REAL*8 ML,K
  LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5

C
  character*(*) OPNRUP
  character*40 STRING
  character*4 dummy

C
  WRITE(6,1100)
  WRITE(6,1110)

C
C*** OPEN THE INPUT FILE
C
  OPEN(UNIT=8,NAME=OPNRUP,TYPE='NEW',
  $  carriagecontrol='list',recordtype='variable')

C
C*** NOW GET THE TITLE BLOCK
C
  WRITE(6,1120)
  WRITE(6,1130)

C
  DO 100 I=1,4
  READ(5,1134) TITLE(I)
  dummy = title(i)
  IF(dummy(1:4) .EQ. POUND(1:4)) GO TO 110
  WRITE(8,1135) TITLE(I)
100 CONTINUE

1 -- sys$desadis:IOT.FOR

```

```

      GO TO 130
C
110 CONTINUE          ! FILL OUT THE BLOCK
    II = I
    DO 120 I = II,4
      TITLE(I) = ' '
      WRITE(8,1135) TITLE(I)
120 CONTINUE
130 CONTINUE
C
c*** Atmospheric parameters:
C
      WRITE(6,1140)
      WRITE(6,1142)
      READ(5,*) U0,Z0,ZR
      WRITE(8,1020) U0,Z0,ZR
C
c*** stability
C
      WRITE(6,1150)
      READ(5,1310) NCHAR,STRING
      istab = 4          ! default is D stability
      IF(STRING.EQ.'A'.OR.STRING.EQ.'a') istab=1
      IF(STRING.EQ.'B'.OR.STRING.EQ.'b') istab=2
      IF(STRING.EQ.'C'.OR.STRING.EQ.'c') istab=3
      IF(STRING.EQ.'D'.OR.STRING.EQ.'d') istab=4
      IF(STRING.EQ.'E'.OR.STRING.EQ.'e') istab=5
      IF(STRING.EQ.'F'.OR.STRING.EQ.'f') istab=6
      goto(161,162,163,164,165,166) istab
161 delta = 0.5          ! A
      beta = 0.9
      m1 = -11.43 * zr**0.103
      sidx_coeff = 0.02
      sidx_pow = 1.22
      sidx_min_dist = 130.
      goto 170
162 delta = 0.33          ! B
      beta = 0.9
      m1 = -25.98 * zr**0.171
      sidx_coeff = 0.02
      sidx_pow = 1.22
      sidx_min_dist = 130.
      goto 170
163 delta = 0.20          ! C
      beta = 0.9
      m1 = -123.4 * zr**0.304
      sidx_coeff = 0.02
      sidx_pow = 1.22
      sidx_min_dist = 130.
      goto 170
164 delta = 0.13          ! D

2 -- sys3desadis:IOT.FOR

```

```

      beta = 0.9
      al = 0.0                ! used for infinity
      sisx_coeff = 0.04
      sisx_pow = 1.14
      sisx_min_dist = 100.
      goto 170
165  delta = 0.1              ! E
      beta = 0.9
      al = 123.4 * zr**0.304
      sisx_coeff = 0.17
      sisx_pow = 0.97
      sisx_min_dist = 50.
      goto 170
166  delta = 0.064           ! F
      beta = 0.9
      al = 25.98 * zr**0.171
      sisx_coeff = 0.17
      sisx_pow = 0.97
      sisx_min_dist = 50.
C
170  WRITE(8,1040) istab
C
172  WRITE(6,1160) delta,beta,al,sisx_coeff,sisx_pow,sisx_min_dist
      read(5,1310) nchar,strings
      if(strings.eq.'d' .or. strings.eq.'D') then
         write(6,1600)
         read(5,*) delta
         goto 172
      else if(strings.eq.'b' .or. strings.eq.'B') then
         write(6,1620)
         read(5,*) beta
         goto 172
      else if(strings.eq.'l' .or. strings.eq.'L') then
         write(6,1660)
         read(5,*) al
         goto 172
      else if(strings.eq.'c' .or. strings.eq.'C') then
         write(6,1670)
         read(5,*) sisx_coeff
         goto 172
      else if(strings.eq.'p' .or. strings.eq.'P') then
         write(6,1680)
         read(5,*) sisx_pow
         goto 172
      else if(strings.eq.'m' .or. strings.eq.'M') then
         write(6,1690)
         read(5,*) sisx_min_dist
         goto 172
      else if(nchar.eq.0 .or. strings.eq.'n' .or. strings.eq.'N') then
         WRITE(8,1020) DELTA,BETA,al
         WRITE(8,1020) sisx_coeff,sisx_pow,sisx_min_dist

```

```

        else
        goto 172
    endif

C
c*** ambient pressure, temperatures, and humidity
C
    write(6,1500)
    read(5,*) tamb,pamb
    tamb = tamb + 273.15      ! K
    vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
    sat = 0.622*vaporp / (pamb- vaporp)
    write(6,1580)
    read(5,1310) nchar,strings
    if(strings.eq.'a' .or. strings.eq.'A') then
        write(6,1585)
        read(5,*) humid
        relhumid = 100.*humid/sat
        write(6,1586) relhumid
        goto 200
    endif
    write(6,1587)
    read(5,*) relhumid
    humid = relhumid/100. * sat
200 rhoa = pamb*(1.+humid)/(.00283+.00456*humid)/tamb
    write(6,1588) rhoa
    write(8,1025) tamb,pamb,humid

C
    isofl = 0
    ihtfl = 0
    htco = 0.
    iwtfl = 0
    wtco = 0.

C
    write(6,2000)
    read(5,1310) nchar,strings
    if(strings.eq.'Y' .or. strings.eq.'y') then
        isofl = 1
        tsurf = tamb
        goto 250
    endif

C
    write(6,2020)
    read(5,1310) nchar,strings
    if(strings.eq.'Y' .or. strings.eq.'y') then
        write(6,2030)
        read(5,*) tsurf
220 write(6,2040)
        read(5,1310) nchar,strings
            if(strings.eq.'V' .or. strings.eq.'v') then
                ihtfl = -1      ! constant value
                write(6,2050)
            endif
    endif

4 -- sys$desadis:IOT.FOR

```

```

        read(5,*) htco
    else if(strings.eq.'C' .or. strings.eq.'c' .or. nchar.eq.0) then
        ihtfl = 1      ! local correlation
    else if(strings.eq.'L' .or. strings.eq.'l') then
        ihtfl = 2      ! LLNL correlation
        htco = 0.0125  ! [=]m/s
        write(6,2043) htco
        read(5,1310) nchar,strings
        if(strings.eq.'Y' .or. strings.eq.'y')
1            read(5,*) htco
            else
            goto 220
        endif

    else
    goto 250
    endif

c
    write(6,2100)
    read(5,1310) nchar,strings
    if(strings.eq.'Y' .or. strings.eq.'y') then
        iwtfl = 1
        write(6,2045)
        read(5,1310) nchar,strings
        if(strings.eq.'V' .or. strings.eq.'v') then
            iwtfl = -1
            write(6,2120)
            read(5,*) wtco
            endif
        endif

c
250 continue
    write(8,1060) isofl,tsurf
    write(8,1060) ihtfl,htco
    write(8,1060) iwtfl,wtco

c
C
c*** sas characteristics
c
    write(6,1510)
    read(5,1415) sas_name
    write(8,1415) sas_name
    sas_mw = 16.04
    sas_temp = 111.7
    sas_rhoe = 1.7928*psmb ! correct to psmb
    sas_cpk = 2730.
    sas_cpe = 1.00
    sas_ufl = 0.15
    sas_lfl = 0.05
    sas_zsp = 0.5
    if(sas_name.eq.'LNG' .or. sas_name.eq.'lng') then
        sas_mw = 16.04

3 -- sys$desadis:IOT.FOR

```

```

sas_temp = 111.7
sas_rhoe = 1.792*pamb ! correct to pamb
sas_cpk = 5.6e-8
sas_cpp = 5.00
sas_ufl = 0.15
sas_lfl = 0.05
sas_zsp = 0.5
endif
if(sas_name.eq.'LPB' .or. sas_name.eq.'lps') then
sas_mu = 44.09
sas_temp = 231.
sas_rhoe = 2.400*pamb ! correct to pamb
sas_cpk = 15.4
sas_cpp = 2.25
sas_ufl = 0.10
sas_lfl = 0.02
sas_zsp = 0.5
endif
270 write(6,1520) sas_mu,sas_temp,sas_rhoe,sas_cpk,sas_cpp,
1 sas_ufl,sas_lfl,sas_zsp
read(5,1310) nchar,string
if(string.eq.'a' .or. string.eq.'M') then
write(6,1550)
read(5,*) sas_mu
goto 270
else if(string.eq.'t' .or. string.eq.'T') then
write(6,1530)
read(5,*) sas_temp
goto 270
else if(string.eq.'d' .or. string.eq.'D') then
write(6,1535)
read(5,*) sas_rhoe
goto 270
else if(string.eq.'h' .or. string.eq.'H') then
write(6,1570)
read(5,*) sas_cpk
goto 270
else if(string.eq.'p' .or. string.eq.'P') then
write(6,1571)
read(5,*) sas_cpp
goto 270
else if(string.eq.'u' .or. string.eq.'U') then
write(6,1572)
read(5,*) sas_ufl
goto 270
else if(string.eq.'l' .or. string.eq.'L') then
write(6,1573)
read(5,*) sas_lfl
goto 270
else if(string.eq.'z' .or. string.eq.'Z') then
write(6,1574)

```

```

        read(5,*) sas_zsp
        goto 270
    else if(nchar.eq. 0 .or. string.eq.'n' .or. string.eq.'N') then
        WRITE(8,1020) sas_mv, sas_tmpr,sas_rhoe
        write(8,1020) sas_cpk,sas_cpp
        WRITE(8,1020) sas_ufl,sas_lfl,sas_zsp
    else
        goto 270
    endif

C
C density curve if isothermal
C
    if(isofl .eq. 0) goto 460
    WRITE(6,1161)
    WRITE(6,1162)
    WRITE(6,1163)
    WRITE(6,1164) rhoa
    WRITE(6,1165)
    WRITE(6,1166)
    goto 320

C
280 write(6,1290)
C
320 LUNIN = 5
    WRITE(6,1300)
    READ(5,1310) NCHAR,STRING
    IF(STRING.EQ.'y' .or. string.eq.'Y') GO TO 360
    GO TO 400
360 WRITE(6,1320)
    READ(5,1310) NCHAR,STRING
    OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=280)
    LUNIN = 10
400 CONTINUE
    IF(LUNIN .EQ. 5) WRITE(6,1170) isen
    READ(LUNIN,*) NP
    WRITE(8,1040) NP
    IF(LUNIN .EQ. 5) WRITE(6,1180)

C
    DO 440 I=1,NP
        den(4,i) = 0.                ! 0.0 by default for isotherm
        den(5,i) = tamb              ! tamb for isotherm
        READ(LUNIN,*) DEN(1,I),DEN(2,I),DEN(3,I)
        if(i .eq.1 .and.
1          (den(3,1)/rhoa.st.1.005 .or. rhoa/den(3,1).st.1.005)) then
            den(3,i) = rhoa
            write(6,1340) rhoa
        endif
        if(i.eq.np) THEN
            IF(      den(2,i)/sas_rhoe .st. 1.005
1              .or.  sas_rhoe/den(2,i) .st. 1.005
1              .or.  den(3,i)/sas_rhoe .st. 1.005

```

7 -- systdesadis:IOT.FOR



```

1      .or.    sas_rhoe/den(3,i) .gt. 1.005) then
          den(2,i) = sas_rhoe
          den(3,i) = sas_rhoe
          write(6,1341) sas_rhoe
          endif
          endif
WRITE(8,1025) DEN(1,I),DEN(2,I),DEN(3,I),Den(4,I),den(5,i)
440 CONTINUE
IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
C
460 WRITE(6,1280)
READ(5,*) CcLOW
if(cclow .le. 0.) cclow=0.005    ! don't let 0. set through
WRITE(8,1010) CcLOW
C
C
C*** source description
C
      write(6,1440)
      read(5,*) smass0
      write(8,1020) smass0
C
C
      check4 = .false.
      write(6,1400)
      read(5,1410) dummy
      if(dummy.eq.'y' .or. dummy.eq.'Y') goto 480
      goto 520
480 continue
      write(6,1420)
      read(5,*) ess
      write(6,1430)
      read(5,*) riss
      nr = 4
C
      tend = 6023. ! [=] sec
C
      et(1,1) = 0.
      et(2,1) = ess
      rit(2,1)= riss
      et(1,2) = tend
      et(2,2) = ess
      rit(2,2)= riss
      et(1,3) = tend + 1.
      et(2,3) = 0.
      rit(2,3)= 0.
      et(1,4) = tend + 2.
      et(2,4) = 0.
      rit(2,4)= 0.
      slen = sqrt(pi*riss**2)

3 -- sys$desadis:IOT.FOR

```

```

      suid = slen
      check4 = .true.      ! steady state run
      goto 760
C
520 continue
C
      WRITE(6,1190)
      WRITE(6,1200)
      WRITE(6,1210)
      WRITE(6,1220)
      WRITE(6,1165)
      WRITE(6,1230)
      WRITE(6,1240)
      WRITE(6,1250)
      goto 600
C
560 write(6,1290)
C
600 LUNIN = 5
      WRITE(6,1330)
      READ(5,1310) NCHAR,STRING
      IF(STRING.EQ.'Y' .OR. string.EQ.'y') goto 640
      goto 680
640 WRITE(6,1320)
      READ(5,1310) NCHAR,STRING
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=560)
      LUNIN = 10
680 CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1260) isen
      READ(LUNIN,*) NP
      IF(LUNIN .EQ. 5) WRITE(6,1270)
C
      DO 720 I=1,NP
      READ(LUNIN,*) ET(1,I),ET(2,I),R1T(2,I)
720 CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
760 continue
      WRITE(8,1040) NP
      DO 800 I=1,NP
800 WRITE(8,1030) ET(1,I),ET(2,I),R1T(2,I)
C
      if(et(2,1).eq.0. .and. smass0.ne.0.) check2=.true. ! HSE type spill
      write(8,*) check1,check2,again,check3,check4,check5
C
      istat = lib$date_time(tinp)
      WRITE(8,1050) TINP
C
      if(check4) write(8,1020) ess,slen,suid      ! steady state
C
9 -- sys$desadis:IOT.FOR

```

CLOSE(UNIT=8)

C

C

```
1010 format(1x,1p14.7)
1020 format(3(1x,1p14.7))
1025 format(5(1x,1p14.7))
1030 format(1x,2(1p14.7,1x),1p14.7)
1040 format(1x,i4)
1050 format(a24)
1060 format(1x,i4,1x,1p14.7)
```

C

```
1100 FORMAT(5X,'INPUT MODULE -- DEGADIS MODEL')
1110 FORMAT(/,5X,'*****')
1120 FORMAT(5X,'Enter Title Block -- up to 4 lines of 80',
  $' characters')
1130 FORMAT(5X,'To stop, type '//')
1134 FORMAT(A80)
1135 FORMAT(A80)
1140 FORMAT(5X,'ENTER WIND PARAMETERS -- U0 (m/s), Z0 (m), ',
  $'and ZR(m)')
1142 format(5x,'U0 -- Wind velocity at reference height Z0',
  $/,5x,'ZR -- Surface Roughness')
1150 FORMAT(/,5X,'Enter the Pasquill stability class: (A,B,C,',
  $'D,E,F) <D> ', $)
1160 format(/,' The values for the atmospheric parameters',
  $' are set as follows:',
  $/, ' DELTA: ',F12.4,
  $/, ' BETA: ',F12.4,
  $/, ' Monin-Obukhov length: ',F12.4,' m',
  $/, ' Sigma X Coefficient: ',F12.4,
  $/, ' Sigma X Power: ',F12.4,
  $/, ' Sigma X Minimum Distance: ',F12.4,' m',
  $/, ' Do you wish to change any of these?',
  $/, ' (No,Delta,Beta,Length,Coefficient,Power,Minimum) <N> ', $)
1161 FORMAT(/,5X,'The density is determined as a function of con',
  $'centration')
1162 FORMAT(5X,'by a listing of ordered pairs supplied by the user')
1163 FORMAT(5X,'Use the following form:')
1164 FORMAT(/,5X,5X,'first point',6X,'-- pure air y=0.0,Cc=0.,',
  1 ' RHOG=RHOA ',1p13.5)
1165 FORMAT(3(15X,'.',/))
1166 FORMAT(5X,5X,'last point',7x,'-- pure gas y=1.0,Cc=RHOE',
  1 ' RHOG=RHOE')
1170 FORMAT(/,5X,'ENTER THE NUMBER OF DATA TRIPLES (max=',i2,')',
  $' FOR THE DENSITY FUNCTION: ', $)
1180 FORMAT(/,5X,'Enter Mole frac, Cc (kg/m**3), then RHOG ',
  1 '(kg/m**3) by triples')
```

C

C

```
1190 FORMAT(/,5X,10X,'Source Description')
1200 FORMAT(1X,/,5X,'The same form used by the density description')
```

10 -- sys\$degadis:IOT.FOR

```

1210 FORMAT(5X,'is used by the source description as follows')
1220 FORMAT(/,5X,5X,'first point',6X,'-- time=0 E,R1 at initial ',
      $'(nonzero) values')
1230 FORMAT(5X,5X,'nxt to last point -- time=TEND E,R1=0.')
1240 FORMAT(5X,5X,'last point',6X,'time=TEND+ E,R1=0.')
1250 FORMAT(/,5X,'Note: the final time is the last time entered ',
      $'where E and R1 are non-zero')
1260 FORMAT(/,5X,'ENTER THE NUMBER OF TRIPLES (max= ',i2,') FOR ',
      $'THE SOURCE DESCRIPTION: ',i)
1270 FORMAT(/,5X,'Enter TIME (sec), EVOLUTION RATE (ks/m**3), ',
      $'and POOL RADIUS (m)')
1280 FORMAT(5X,'Enter the LOWEST CONCENTRATION OF INTEREST (ks/',
      $'m**3) : ',i)
1290 format(/,' This file was not found.')
1300 FORMAT(/,' Do you have an input file for the Density ',
      $'function? [y or N] ',i)
1310 FORMAT(Q,A20)
1320 FORMAT(' Enter the file name: [DIR]FILE_NAME.EXT ',i)
1330 FORMAT(' Do you have an input file for the Source ',
      $'Description? [y or N] ',i)
1340 format(/,' Air density corrected to ',l$13.5' ks/m**3',/)
1341 format(/,' Contaminant density corrected to ',l$13.5' ks/m**3',/)

```

C

C

```

1400 format(/,' Is this a Steady state simulation? <y or N> ',i)
1410 format(a4)
1415 format(a3)
1420 format(/,' Enter the desired evolution rate [=] ks/sec : ',i)
1430 format(' Enter the desired source radius [=] m : ',i)
1440 format(/,' Specification of source rate and extent.',
      $'///,' Enter the initial mass of pure gas',
      $' over the source. (ks)'/,' (Positive or zero): ',i)

```

C

C

```

1500 format(/,' Enter the ambient temperature(C) and pressure',
      1 '(atm): ',i)
1510 format(/,' Enter the code name of the diffusing species: ',i)
1520 format(/,' The characteristics for the gas are set as follows:',//
      $' Molecular weight: ',f7.2,/,
      $' Storage temperature [K]: ',l$13.5,/,
      $' Density at storage temperature, PAMB [ks/m**3]: ',l$13.5,/,
      $' Mean Heat capacity constant ',l$13.5,/,
      $' Mean Heat capacity power ',l$13.5,/,
      $' Upper Flammability Limit [mole frac] ',l$13.5,/,
      $' Lower Flammability Limit [mole frac] ',l$13.5,/,
      $' Height of Flammability Limit [m] ',l$13.5,/,
      $' Do you wish to change any of these? ',
      $'(No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z)',
      $' <N> ',i)
1530 format(' Enter the desired Storage Temperature: ',i)
1535 format(' Enter the desired Density at Storage ',

```

```

1      'Temperature and', ' ambient pressure: ',%)
1550 format(' Enter the desired Molecular Weight: ',%)
1570 format(' Enter the desired Mean Heat Capacity constant: ',%)
1571 format(' Enter the desired Mean Heat Capacity power: ',%)
1572 format(' Enter the desired Upper Flammability Limit: ',%)
1573 format(' Enter the desired Lower Flammability Limit: ',%)
1574 format(' Enter the desired Height for the flammable limit calcula',
1      'tions: ',%)
1580 format(/, ' The ambient humidity can be entered as Relative ',
1      ' or Absolute.',/, ' Enter either R or A <R or a>: ',%)
1585 format(' Enter the absolute humidity (kg water/kg BDA): ',%)
1586 format(' This is a relative humidity of ',1e13.5,' %')
1587 format(' Enter the relative humidity (X): ',%)
1588 format(/, ' Ambient Air density is ',1e13.5,' kg/m**3')
C
C
1600 format(' Enter the desired DELTA: ',%)
1620 format(' Enter the desired BETA: ',%)
1660 format(' Note: For infinity, ML = 0.0',/,
$      ' Enter the desired Monin-Obukhov length: (a) ',%)
1670 format(' Enter the desired Sigma X Coefficient: ',%)
1680 format(' Enter the desired Sigma X Power: ',%)
1690 format(' Enter the desired Sigma X Minimum distance: (a) ',%)
C
2000 format(/, ' Is this an Isothermal spill? <y or N> ',%)
2020 format(/, ' Is heat transfer to be included in the',
1      ' calculations <y or N> ',%)
2030 format(' Enter the surface temperature [=] K : ',%)
2040 format(' Do you want to use the built in correlation,',
1      ' the LLNL correlation, or',/, ' enter',
1      ' a particular value?',/,
1      ' (Corr,LLNLcorr,Value) <C> ',%)
2043 format(/, ' The form of the correlation is:',/,
1      ' Q = (Vh * rho * cp) * area * (tsurf-temp)',/,
1      ' with Vh = ',1e13.5,' m/s.',/,
1      ' Do you wish to change the value of Vh? (y or N): ',%)
2045 format(' Do you want to use the built in correlation or enter',
1      ' a particular value?',/, ' <C or v> ',%)
2050 format(' Enter the HT coefficient value [=] J/m**2/s/K : ',%)
2100 format(/, ' Is water transfer to be included in the',
1      ' source <y or N> ',%)
2120 format(' Enter the WT coefficient value [=] kmol/m**2/s/atm : ',%)
C
C
      RETURN
      END
###

```

```

C.....
C
C SUBROUTINE FOR SOURCE EVALUATION WHEN NO GAS BLANKET
C IS PRESENT.
C
C SUBROUTINE NOBL(timeout)
C
C Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C include 'sys$desadis:DEGADIS1.dec'
C
C COMMON
C /GEN1/ ET(2,isn),R1T(2,isn)
C /ERROR/ STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtyc,wteb,wtab,wtuh,XLI,
C $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srccs,srccut,
C $ htcut,ERNOBL,NOBLpt,crfser,epsilon
C /PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C /comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wtco
C /PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C /com_ss/ ess,slen,swid,outcc,outcz,outb,outl,swcl,swal,senl,srhl
C /phlas/ check1,check2,again,check3,check4,check5
C /ALP/ ALPHA,alphal
C /phicom/ iphifl,dellay
C
C REAL*8 ML,K
C
C LOGICAL REV
C logical check1,check2,again,check3,check4,check5
C DATA REV/.TRUE./
C REAL*8 L
C
C data h/0./,Ri/0./
C data delt_min/0.5/
C
C DELTAT = (TEND - TSC1)/FLOAT(NOBLPT)
C if(deltat .lt. delt_min) then
C     noblpt = int((tend-tsc1)/delt_min) +1
C     deltat = (tend-tsc1)/float(noblpt)
C endif
C
C TO = TSC1
C IF(DELTAT .LT. 2.) GO TO 100
C
C WRITE(lunlos,1100)
C WRITE(lunlos,*) DELTAT
C 1100 FORMAT(5X,'TIME INCREMENT USED ON LAST PORTION OF SOURCE CALC')
C
C 100 CONTINUE
C
C ESTABLISH LOOP TO FINISH SOURCE
C
C 1 -- sys$desadis:NOBL.FOR

```

```

C
DO 110 I = 1,NOBLPT
C
TIME = T0 + FLOAT(I)*DELTAT
IF(I .EQ. NOBLPT) TIME = TEND
L = SQRTPI*AFGEN(R1T,TIME,'R1T-BL')
erate = AFGEN(ET,TIME,'ET-BL')
flux = Erate/L/L
C
qstar = rhoe * k*ustar*alpha1*dellay/(dellay-1.)/phihat(rhoe,L)
if(abs(flux/qstar) .gt. ernobl) then
    check3 = .true.
    timeout = time
    return
end if
C
call szf(flux,L,sz,cclay,wclay,rholay)
cc = cclay*dellay
C
call adiab(0,wc,wc,yc,yc,cc,rho,wc,enthalpy,temp)
C
IF(Erate .LT. EMAX) GO TO 220
EMAX = Erate
RM = AFGEN(R1T,TIME,'R1T-BL')
SZH = SZ
220 CONTINUE
RLIST = AFGEN(R1T,TIME,'R1T-BL')
RMAX = dMAX1(RMAX,RLIST)
C
WRITE(9,2000) TIME,RLIST,h,flux,SZ,yc,yc,rho,Ri,wc,wc,enthalpy,temp
C
if(i.eq.5 .and. check4) goto 500      ! steady
C
110 CONTINUE
RETURN
C
C
500 continue                        ! steady state completion
outcc = cc
swcl = wc
swal = wa
senl = enthalpy
srhl = rho
outsz = sz
outl = sqrt(pi) * rlist
outb = outl/2.
return
2000 format(1p16.9,1x,1p16.9,<iout_src-2>(1x,1p13.6))
END
####

```

```

C.....
C
C   SUBROUTINES OB AND OBOUT ARE USED IN THE OBSERVER INTEGRATIONS
C   OVER THE SOURCE.
C
C   SUBROUTINE OB(time,Y,D,PRMT)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       include 'sys$desadis:DEGADIS2.dec'
C
C       COMMON
C       $/GEN3/ rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
C       $ srcwa(2,max1),srcenth(2,max1)
C       $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C       $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wteo
C       $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C       $/ALP/ALPHA,alpha1
C       $/phicom/ iphifl,dellay
C
C       REAL*8 K,ML
C       logical flas
C
C       DIMENSION Y(1),D(1),PRMT(1)
C       INTEGER HWIDTH,Hrate,Crate,BDArate,Hrate
C       DATA HWIDTH/1/,Hrate/2/,Crate/3/,BDArate/4/,Hrate/5/
C
C*** PASS TO IN PRMT(6)
C
C       flas = isofl.ea. 1 .or. ihtfl.ea. 0
C
C       T01 = PRMT(6)
C       :UP = prmt(7)
C       XI = XIT(TIME,T01)
C       RG = AFGEN(RADG,TIME,'RADG')
C       RLEN = PRMT(13)
C
C       BIPR = 0.
C       IF((ABS(XI)-RG)/RG .GE. 0.01) WRITE(lunlog,1000) XI,RG
C       IF(ABS(XI) .LT. RG) BIPR = sqrt(RG*RG - XI*XI)
C
C       UI = UIT(TIME,T01)
C
C       Q = AFGEN(QSTR,TIME,'QSTR')
C       wc = AFGEN(srcwc,time,'srcwc')
C       wa = AFGEN(srcwa,time,'srcwa')
C       enth = AFGEN(srcenth,time,'srcenth')
C
C       wclay = Y(Crate)/Y(Hrate)
C       walay = Y(BDArate)/Y(Hrate)
C       if(.not.flas) enthlay = Y(Hrate)/Y(Hrate)
C
C 1 -- sys$desadis:OB.FOR

```



```

C      call tprop(1,wclay,walay,enthlay,yc,ys,ws,temp,rholay,cp)
      cclay = wclay*rholay
C
      prnt(8) = cclay
      prnt(9) = wclay
      prnt(10) = walay
      prnt(11) = enthlay
      prnt(12) = rholay
C
      cc = cclay*dellay
      rho = dellay*(rholay-rhoa) + rhoa
C
      szob = 0.01
      ars = Q*(xi-xup)/cc/(u0*z0/alpha1)
      if((xi.gt. xup .and. ars.gt.0.)
      1 szob = ars**(1./alpha1) * z0
C
      HEFF = GAMMAF/ALPHA1* SZOB
      RISTR=RIF(RHO,HEFF)
      PHI = PHIF(RISTR,0.)
      welay = dellay * K*USTAR* ALPHA1/PHI
C
      D(HWIDTH)= UI * BIPR / RLEN
      D(Crate) = D(HWIDTH)*RLEN * Q
      D(Hrate) = (Q/wc + rhoa*welay) * D(HWIDTH)*RLEN
      D(BDArate) = (2*wa/wc + rhoa*welay/(1.+humid)) * D(HWIDTH)*RLEN
      if(flag) return
      D(Hrate) = Q * enth/wc * D(HWIDTH)*RLEN
C
      1000 FORMAT(' ?0B? -- Value of XI ',1pG13.4,'; Value of RG ',
      $ 1pG13.4)
      RETURN
      END
C
C
C      SUBROUTINE OBOUT( X, Y, DERY, IHLF, NDIM, PRMT)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      DIMENSION X(1), Y(1), DERY(1), PRMT(1)
C
      PRMT(14) = prnt(8)      ! cclay
      PRMT(15) = prnt(9)      ! wclay
      PRMT(16) = prnt(10)     ! walay
      PRMT(17) = prnt(11)     ! enthlay
      PRMT(18) = prnt(12)     ! rholay
      RETURN
      END

```

\*\*\*\*

```

C.....
C
C   FUNCTION PSI
C
C*** AS PER COLENBRANDER --
C
C*** THIS FUNCTION HAS BEEN DERIVED FROM BUSINGER,J.A.
C*** WORKSHOP ON MICROMETEOROLOGY, CHAPTER 2, HAUGEN,D.A. (ED.)
C*** AMERICAN METEOROLOGICAL SOCIETY.
C
C   FUNCTION PSIF(Z,ML)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sys$desadis:DEGADIS1.dec'
C
C   REAL*8 ML
C
C   IF( ML ) 10,20,30
C
C 10  A = (1.-15.*Z/ML)**.25
C     PSIF = 2.*dLOG((1.+A)/2.) + dLOG((1.+A*A)/2.) - 2.*dATAN(A) +
C     $ PI/2.
C     RETURN
C
C 20  PSIF = 0.
C     RETURN
C
C 30  PSIF = -4.7*Z/ML
C     RETURN
C     END
****

```

```

C.....
C
C   SUBROUTINES FOR PSEUDO-STEADY STATE INTEGRATION.
C
C   SUBROUTINE PSS(DIST,Y,DERY,PRMT)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       include 'sys$desadis:DEGADIS2.dec/list'
C
C       parameter (zero=1.D-10, rcrit=2.D-3)
C
C       COMMON
C       $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C       $/consta/ istab,tamb,psab, humid, isofl,tsurf,ihtfl,htco,iutfl,utco
C       $/ALP/ ALPHA,alpha1
C       $/phicom/ i=hi1,dellay
C       $/srd_con/ ce, delrhomin
C
C       REAL*8 K,ML
C
C       DIMENSION Y(1),DERY(1),PRMT(1)
C       DATA rhoth/1/,SY/2/,BEFF/3/,dh/4/
C       INTEGER rhoth,SY,BEFF,dh
C
C*** PRMT I/O SETUP
C*** I      VALUE      IN/OUT
C*** ----
C*** 6      E          IN
C*** 7      Cc         OUT
C*** 8      B          OUT
C*** 9      CON DERY(BEFF)  IN
C*** 10     CON DERY(SZ)   IN
C*** 11     NREC(I,1)      OUT -- STARTS OUTPUT COUNTER
C*** 12     DIST          OUT
C*** 13
C*** 14     yc           out
C*** 15     rho          out
C*** 16     temp         out - if recorded
C*** 17     sanna        out - if recorded
C*** 18
C*** 19
C*** 20
C*** 21     sz
C*** 22     sz
C
C       Erate = PRMT(6)
C       B = Y(BEFF) - SQRTPI/2.*Y(SY)
C
C
C   -- sys$desadis:PSS.FOR

```

```

c using the last value for Sz
c
      sz0 = prmt(22)
      sz = sz0
c
c*** MATERIAL BALANCE
c
      iii = 0
100  Cc = Erate*ALPHA1/2./U0*(Z0/SZ)**ALPHA/SZ/Y(BEFF)
      call adiabat(0,wc,wa,wc,va,cc,rho,ua,enth,temp)
      cclay = cc/dellay
      call adiabat(0,wc,wa,cclay,va,cclay,rholay,uwl,enth,temlay) ! for wa
      call addheat(cclay,y(dh),rholay,temlay,cp)
      prod = dmax1( Y(rhouh)/rholay/prmt(19), zero)
      sz = ( prod )**(1./alpha1) * z0
      dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
      if(dif .gt. rcrit) then
          sz0 = sz
          iii=iii+1
          if(iii .gt. 20) call trap(32)
          goto 100
      endif
      prmt(20) = rholay
      prmt(21) = sz
      HEFF = GAMMAF/ALPHA1*SZ
c
      rit = 0.
      temp = temlay
      if(isofl.eq.0 .or. ihtfl.ne.0) then
          rho = dellay*(rholay-rhoa) + rhoa      ! estimate
          temp = (ua/rho)*(rholay*temlay/uwl)    ! estimate
          rit = rift(temp,heff)
      endif
      RISTR = RIF(RHO,HEFF)
      PHI = PHIF(RISTR,rit)
c
c*** CALCULATE DERIVATIVES
c
      DERY(BEFF) = 0.
      delrho = rho-rhoa
      IF(delrho .GT. delrho.in) DERY(BEFF) = PRMT(9)*sqrt(delrho/rhoa)
      $      *(SZ/Z0)**(.5 - ALPHA)
c
      DERY(SY) = 4.*BETA/PI/Y(SY)*Y(BEFF)**2 *
      $      (DELTA*SQPI02/Y(BEFF)) ** (1./BETA)
c
c
      heish = heff*dellay
      call surface(temlay,heish,rholay,uwl,cp,watrtc,arte)
      if(temp.ge. tsurf .or. temlay.ge. tamb) arte = 0.
      rhoub = rholay* prmt(19) * (sz/z0)**alpha1 * Y(beff)

```

```

d_rhouhb = prmt(19)*y(beff)/phi
dERY(dh) = (arte*y(beff)/dellay - Y(dh)*d_rhouhb)/rhouhb
dERY(rhouh) = (d_rhouhb-Y(rhouh)*dERY(beff))/Y(beff)

```

C

C\*\*\* RETURNED VALUES

C

```

PRMT(7) = Cc
PRMT(8) = B
prmt(14) = yc
prmt(15) = rho
prmt(16) = temp
prmt(17) = (rho1ay-rhoa)/cc1ay      ! samea

```

C

```

RETURN
END

```

###

```

C.....
C
C      SUBROUTINE PSSOUT
C
C      SUBROUTINE PSSOUT(X,Y,D,IHLF,NDIM,PRMT)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'sysdesadis:DEGADIS2.dec'
C
C      parameter (npss=9, zero=1.e-10)
C
C      COMMON
C      $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wteo
C      $/STP/STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C      $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      $/STOPIT/TSTOP
C
C      REAL*8 K,ML
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      DIMENSION Y(1),D(1),PRMT(1),BKSP(npss),OUT(npss),CURNT(npss)
C
C      C*** OUTPUT PARAMETERS
C
C      C*** FROM PSS          OUTPUT TO MODEL
C      C*** -----          -----
C      C*** X                  DIST
C      C*** PRMT(7)            Cc
C      C*** Y(1)              SZ
C      C*** Y(2)              SY
C      C*** PRMT(8)            B
C      C*** PRMT(13)           TO(I)
C      C*** prmt(14)           yc
C      C*** prmt(15)           rho
C      C*** prmt(16)           temp
C      C*** prmt(17)           samsa
C
C      ERM = 0.
C      TSL = TS(PRMT(13),X)
C      prmt(22) = prmt(21)
C      IF(PRMT(11) .NE. 0.) GO TO 90
C
C      C*** STARTUP FOR THE OUTPUT ROUTINE
C
C      RII = -100./STPP
C      RI = 0.
C      CURNT(1) = X
C      curnt(2) = prmt(14)      ! yc
C      CURNT(3) = PRMT(7)      ! cc
C
C      1 -- sysdesadis:PSSOUT.FOR

```

```

      curnt(4) = prmt(15)      ! rho
      curnt(5) = prmt(17)      ! samaa
      curnt(6) = prmt(16)      ! temp
      CURNT(7) = prmt(21)      ! sz
      CURNT(8) = Y(2)          ! sy
      CURNT(9) = PRMT(8)       ! b
      IF(prmt(8) .LE. 0.) CALL trap(16)
90  CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
      IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** STOP INTEGRATION AND GET A NEW OBSERVER WHEN Cc<CcLOW
C
      IF(PRMT(7).GT.CcLOW .OR. TSL.LT.TSTOP) GO TO 95
          if(prmt(11) .lt. 5.) then      ! guarantee 5 records
              era = odlp                 ! force output
              goto 95
          endif
      TSTOP = TSL
      AGAIN = .TRUE.
      GO TO 1000
95  CONTINUE
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
      DO 100 II=1,nps
100  BKSP(II) = CURNT(II)
C
      CURNT(1) = X
      curnt(2) = prmt(14)      ! yc
      CURNT(3) = PRMT(7)       ! cc
      curnt(4) = prmt(15)      ! rho
      curnt(5) = prmt(17)      ! samaa
      curnt(6) = prmt(16)      ! temp
      CURNT(7) = prmt(21)      ! sz
      CURNT(8) = Y(2)          ! sy
      CURNT(9) = PRMT(8)       ! b
C
      RI = RI + 1.
      II = 2                    ! skip DIST,YC; 1 and 2
110  II = II + 1
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
      ERM = dMAX1(ER1,ER2,ERM)
      IF(II .EQ. 3) II = II + 1      ! skip RHO;4
      IF(II .EQ. 7) II = II + 1      ! skip SY;8
      IF(II .EQ. 5) II = II + 1      ! skip TEMP;6
      IF(II .LT. nps) GO TO 110
C
2  -- systdesadis:PSSOUT.FOR

```

C\*\*\* RECORD POINT IF ODLP IS EXCEEDED OR 90 METERS SINCE LAST RECORD  
 C\*\*\* RECORD FIRST POINT

C

DX = CURNT(1) - OUT(1)

IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN

C

C\*\*\* IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE  
 C\*\*\* ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE  
 C\*\*\* LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.

C

DO 120 II=1,npss

IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)

120 OUT(II) = BKSP(II)

C

RI = RII

PRMT(11) = PRMT(11) + 1.

C

WRITE(9,\*) (OUT(II),II=1,npss)

RETURN

C

1000 CONTINUE

C

C\*\*\* STOP INTEGRATION

C

PRMT(12) = X

C

IF(CURNT(1) .EQ. OUT(1)) GO TO 130

C

PRMT(11) = PRMT(11) + 1.

WRITE(9,\*) (CURNT(II),II=1,npss)

C

130 CONTINUE

PRMT(5) = 1.

RETURN

END

####



```

C.....
C
C   SUBROUTINE PSSOUT
C
C   SUBROUTINE PSSOUT(X,Y,DERY,IHLF,NDIM,PRMT)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-M )
C
C   include 'sys$desadis:DEGADIS2.dec/list'
C
C   parameter (npss=9, zero=1.e-10)
C
C   COMMON
C   $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C   $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   $/com_fl/ cflas,cfl1,cufl
C   $/ALP/ALPHA,alpha1
C
C   logical cflas
C   LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C   REAL*8 ML,K
C
C   DIMENSION Y(1),DERY(1),PRMT(1)
C   dimension BKSP(npss),OUT(npss),CURNT(npss)
C
C*** OUTPUT PARAMETERS
C
C*** FROM PSS          OUTPUT TO MODEL
C*** -----          -----
C*** X                  DIST
C*** PRMT(7)            Cc
C*** Y(1)               SZ
C*** Y(2)               SY
C*** PRMT(8)            B
C
C   ERM = 0.
C   prmt(22) = prmt(21)
C
C   IF (PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR THE OUTPUT ROUTINE
C
C   RII = -100./STPP
C   RI = 0.
C   CURNT(1) = X
C   CURNT(2) = PRMT(14)      ! yc
C   CURNT(3) = prmt(7)      ! cc
C   CURNT(4) = prmt(15)     ! rho
C
C 1 -- sys$desadis:PSSOUTSS.FOR

```

```

      CURNT(5) = PRMT(17)      ! sama
      curnt(6) = prmt(16)     ! temp
      curnt(7) = prmt(8)      ! b
      curnt(8) = prmt(21)     ! sz
      curnt(9) = Y(2)         ! sy
C
      90 CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
      IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** STOP INTEGRATION when Cc<CcLOW
C
      IF(PRMT(7).GT.CcLOW) GO TO 95
          if(prmt(11) .lt. 5.) then      ! force output
              era = odlp
              goto 95
          endif
      AGAIN = .TRUE.
      GO TO 1000
      95 CONTINUE
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
      DO 100 II=1,npss
      100 BKSP(II) = CURNT(II)
C
      CURNT(1) = X
      CURNT(2) = PRMT(14)      ! xc
      CURNT(3) = prmt(7)      ! cc
      CURNT(4) = prmt(15)     ! rho
      CURNT(5) = PRMT(17)     ! sama
      curnt(6) = prmt(16)     ! temp
      curnt(7) = prmt(8)      ! b
      curnt(8) = prmt(21)     ! sz
      curnt(9) = Y(2)         ! sy
C
      RI = RI + 1.
      II = 1
      110 II = II + 1
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
      ERM = dMAX1(ER1,ER2,ERM)
      IF(II .EQ. 3) II = 6      ! skip density,sama,temp
      IF(II .LT. npss-1) GO TO 110      ! skip sy
C
C*** RECORD POINT IF ODLP IS EXCEEDED OR 80 METERS SINCE LAST RECORD
C*** RECORD FIRST POINT
C
      DX = CURNT(1) - OUT(1)
C
      2 -- sys$desadis:PSSOUTSS.FOR

```

```

      IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN
C
C*** IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE
C*** ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE
C*** LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.
C
      DO 120 II=1,npss
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(out)
      RETURN
C
1000 CONTINUE
C
C*** STOP INTEGRATION
C
      PRMT(12) = X
C
      IF(RI .EQ. 0.) CALL trap(16)
C
      IF(CURNT(1) .EQ. OUT(1)) GO TO 130
C
      PRMT(11) = PRMT(11) + 1.
      call ssout(out)
C
130 CONTINUE
      PRMT(5) = 1.
C
      RETURN
      END
****

```

```

C.....
C
C   RICHARDSON NUMBER (RI*)
C
C   FUNCTION RIF(RHOG,HEFF)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   COMMON
C   $ /PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C
C   REAL*8 ML,K
C
C   RIF = G*(RHOG-RHOA)/RHOA*HEFF/USTAR/USTAR
C
C   RETURN
C   END
C.....
C
C   RICHARDSON NUMBER (Rit)
C
C   FUNCTION RIFt(temp,HEFF)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   COMMON
C   $ /PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtf,wtco
C   $/alp/ alpha,alpai
C
C   REAL*8 ML,K
C
C   wind = u0*(heff/z0)**alpha
C   RIFt = dmax(1,(G*(tsurf-temp)/temp*HEFF/USTAR/wind,0.00)
C
C   RETURN
C   END
C.....
C
C   PHI FUNCTION
C
C   FUNCTION PHIF(RI,rit)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   common /phicom/ iphifl,delay
C
C   1 -- sys$desadis:RIPPHIF.FOR

```

```

C
    phif= 0.
    soto(10,1000,2000,3000,9000),i=phif1
    soto 9000
C
    10 IF(RI) 100,200,300
C
    100 PHIF = 0.74/(1. + 0.65*ABS(RI)**.6)
    RETURN
C
    200 PHIF = 0.74
    RETURN
C
    300 PHIF = 0.74 + 0.25*(RI)**0.7 + 1.2E-7*RI*RI*RI
    RETURN
C
C
    1000 IF(RI) 1100,1200,1300
C
    1100 PHIF = 0.88/(1. + 0.65*ABS(RI)**.6)
    RETURN
C
    1200 PHIF = 0.88
    RETURN
C
    1300 PHIF = 0.88 + 9.9e-2*(RI)**1.04 + 1.4E-25*RI**5.7
    RETURN
C
C
C
    2000 corrl = 0.25* rit**.666666 + 1.
    corr = sarr(corrl)
    riw = ri/corrl
    IF(RI) 2100,2200,2300
C
    2100 PHIF = 0.88/(1. + 0.65*ABS(RIw)**.6)/corr
    RETURN
C
    2200 PHIF = 0.88/corr
    RETURN
C
    2300 PHIF = (0.88 + 9.9e-2*(RIw)**1.04 + 1.4E-25*RIw**5.7)/corr
    RETURN
C
C
    3000 corrl = 0.25* rit**.666666 + 1.
    corr = sarr(corrl)
    riw = ri/corrl
    IF(RI) 3100,3200,3300
C
    3100 PHIF = 0.88/corr
C
2 -- sys$desadis:RIPHIF.FOR

```

```

      RETURN
C
3200 PHIF = 0.88/corr
      RETURN
C
3300 PHIF = (0.88 + 9.9e-2*(RIW)**1.04 + 1.4E-25*RIW**5.7)/corr
      RETURN
C
C
9000 call trap(29)
      return
      END
C
C
C.....
C
C
      function phihat(rho,fetch)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      common
      $/parm/ u0,z0,zr,m1,ustar,k,s,rhoc,rhoa,delta,beta,gammaf,cclow
      $/alp/ alpha,alpha1
      $/phicom/ phic,dellay

C
      real*8 k,m1

C
      data phic/3.1/

C
      if(rho .le. rhoa) then
         phihat = 0.88
         return
      endif

C
      pow = 1./alpha1
      p1 = 1.04/alpha1
      Ci = s*(rho-rhoc)/rhoa*z0/ustar**2*gammaf/alpha1
      Ci = Ci * (k*ustar*alpha1**2 /u0/z0/ phic*dellay/(dellay-1.)) ** pow

C
      Cip = 0.099*Ci**1.04

C
      phihat = dlog((.88+ Cip*fetch**p1)/.88)/Cip/fetch
      phihat = 1./phihat

C
      return
      end
****

3 -- sysidesadis:RIPHIF.FOR

```

```

C .....
C
C SUBROUTINE RKGST
C .....
C
C This routine was originally supplied by Digital Equipment
C Corporation as part of the Scientific Subroutine Package
C available for RT-11 as part of the Fortran Enhancement
C Package. It was upgraded for use as the integration
C routine in this package.
C .....
C
C PURPOSE
C TO SOLVE A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL
C EQUATIONS WITH GIVEN INITIAL VALUES.
C
C USAGE
C CALL RKGST (PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C PARAMETERS FCT AND OUTP REQUIRE AN EXTERNAL STATEMENT.
C
C DESCRIPTION OF PARAMETERS
C
C PRMT AN INPUT AND OUTPUT VECTOR WITH DIMENSION GREATER
C OR EQUAL TO 5, WHICH SPECIFIES THE PARAMETERS OF
C THE INTERVAL AND OF ACCURACY AND WHICH SERVES FOR
C COMMUNICATION BETWEEN SUBROUTINES OUTP AND FCT
C (FURNISHED BY THE USER) AND SUBROUTINE RKGST.
C EXCEPT PRMT(5) THE COMPONENTS ARE NOT DESTROYED
C BY SUBROUTINE RKGST AND THEY ARE:
C PRMT(1) LOWER BOUND OF THE INTERVAL (INPUT),
C PRMT(2) UPPER BOUND OF THE INTERVAL (INPUT),
C PRMT(3) INITIAL INCREMENT OF THE INDEPENDENT VARIABLE
C (INPUT),
C PRMT(4) UPPER ERROR BOUND (INPUT). IF RELATIVE ERROR IS
C GREATER THAN PRMT(4), INCREMENT GETS HALVED.
C IF RELATIVE ERROR LESS THAN PRMT(4)*EXPAND,
C INCREMENT GETS DOUBLED.
C THE USER MAY CHANGE PRMT(4) BY MEANS OF HIS
C OUTPUT SUBROUTINE.
C PRMT(5) MAXIMUM STEP SIZE ORDER OF MAGNITUDE (INPUT).
C SUBROUTINE RKGST INITIALIZES
C PRMT(5)=0. IF THE USER WANTS TO TERMINATE
C SUBROUTINE RKGST AT ANY OUTPUT POINT, HE HAS TO
C CHANGE PRMT(5) TO NON-ZERO BY MEANS OF SUBROUTINE
C OUTP. FURTHER COMPONENTS OF VECTOR PRMT ARE
C FEASIBLE IF ITS DIMENSION IS DEFINED GREATER
C THAN 5. HOWEVER SUBROUTINE RKGST DOES NOT REQUIRE
C AND CHANGE THEM. NEVERTHELESS THEY MAY BE USEFUL
C FOR HANDING RESULT VALUES TO THE MAIN PROGRAM

```

C (CALLING RKGST) WHICH ARE OBTAINED BY SPECIAL  
 C MANIPULATIONS WITH OUTPUT DATA IN SUBROUTINE OUTP.  
 C Y INPUT VECTOR OF INITIAL VALUES. (DESTROYED)  
 C LATER, Y IS THE RESULTING VECTOR OF DEPENDENT  
 C VARIABLES COMPUTED AT INTERMEDIATE POINTS X.  
 C DERY INPUT VECTOR OF ERROR WEIGHTS. (DESTROYED)  
 C ERROR WEIGHTS ARE CENTERED AT ONE. IF ONE PARA-  
 C METER NEEDS A TIGHTER ERROR CRITERIA, THE WEIGHT IS  
 C GREATER THAN ONE. IF A PARAMETER NEED NOT BE DETER-  
 C MINED SO PRECISELY, THE WEIGHT SHOULD BE LESS  
 C THAN ONE. IN OTHER WORDS,  
 C 
$$\text{ERROR CRITERIA}(I) = \text{PRMT}(4) / \text{WEIGHT}(I)$$
  
 C WHERE I IS THE SUBSCRIPT OF A DEPENDENT VARIABLE.  
 C LATER, DERY IS THE VECTOR OF DERIVATIVES, WHICH  
 C BELONG TO FUNCTION VALUES Y AT A POINT X.  
 C NDIM AN INPUT VALUE, WHICH SPECIFIES THE NUMBER OF  
 C EQUATIONS IN THE SYSTEM.  
 C IHLF AN OUTPUT VALUE, WHICH SPECIFIES THE NUMBER OF  
 C BISECTIONS OF THE INITIAL INCREMENT. IF IHLF BE-  
 C COMES GREATER THAN 10, SUBROUTINE RKGST RETURNS THE  
 C ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM. ERROR  
 C MESSAGE IHLF=12 OR IHLF=13 APPEARS IN CASE  
 C  $\text{PRMT}(3)=0$  OR IN CASE  $\text{SIGN}(\text{PRMT}(3)) \neq \text{SIGN}(\text{PRMT}(2) - \text{PRMT}(1))$  RESPECTIVELY.  
 C FCT THE NAME OF AN EXTERNAL SUBROUTINE USED. THIS  
 C SUBROUTINE COMPUTES THE RIGHT HAND SIDES DERY OF  
 C THE SYSTEM TO GIVEN VALUES X AND Y. ITS PARAMETER  
 C LIST MUST BE X,Y,DERY,PRMT. SUBROUTINE FCT SHOULD  
 C NOT DESTROY X AND Y.  
 C OUTP THE NAME OF AN EXTERNAL OUTPUT SUBROUTINE USED.  
 C ITS PARAMETER LIST MUST BE X,Y,DERY,IHLF,NDIM,PRMT.  
 C NONE OF THESE PARAMETERS (EXCEPT, IF NECESSARY,  
 C  $\text{PRMT}(4), \text{PRMT}(5), \dots$ ) SHOULD BE CHANGED BY  
 C SUBROUTINE OUTP. IF  $\text{PRMT}(5)$  IS CHANGED TO NON-ZERO,  
 C SUBROUTINE RKGST IS TERMINATED.  
 C AUX AN AUXILIARY STORAGE ARRAY WITH 8 ROWS AND NDIM  
 C COLUMNS.  
 C  
 C REMARKS  
 C  
 C THE PROCEDURE TERMINATES AND RETURNS TO CALLING PROGRAM, IF  
 C (1) MORE THAN 10 BISECTIONS OF THE INITIAL INCREMENT ARE  
 C NECESSARY TO GET SATISFACTORY ACCURACY (ERROR MESSAGE  
 C IHLF=11),  
 C (2) INITIAL INCREMENT IS EQUAL TO 0 OR HAS WRONG SIGN  
 C (ERROR MESSAGES IHLF=12 OR IHLF=13),  
 C (3) THE WHOLE INTEGRATION INTERVAL IS WORKED THROUGH,  
 C (4) SUBROUTINE OUTP HAS CHANGED  $\text{PRMT}(5)$  TO NON-ZERO.  
 C  
 C SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED  
 C THE EXTERNAL SUBROUTINES FCT(X,Y,DERY,PRMT) AND



C     OUTP(X,Y,DERY,IHLF,NDIM,PRMT) MUST BE FURNISHED BY THE USER.

C     METHOD

C     EVALUATION IS DONE BY MEANS OF FOURTH ORDER RUNGE-KUTTA  
 C     FORMULAE IN THE MODIFICATION DUE TO GILL. ACCURACY IS  
 C     TESTED COMPARING THE RESULTS OF THE PROCEDURE WITH SINGLE  
 C     AND DOUBLE INCREMENT.  
 C     SUBROUTINE RKGST AUTOMATICALLY ADJUSTS THE INCREMENT DURING  
 C     THE WHOLE COMPUTATION BY HALVING OR DOUBLING. IF MORE THAN  
 C     10 BISECTIONS OF THE INCREMENT ARE NECESSARY TO GET  
 C     SATISFACTORY ACCURACY, THE SUBROUTINE RETURNS WITH  
 C     ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM.  
 C     TO GET FULL FLEXIBILITY IN OUTPUT, AN OUTPUT SUBROUTINE  
 C     MUST BE FURNISHED BY THE USER.  
 C     FOR REFERENCE, SEE  
 C     RALSTON/WILF, MATHEMATICAL METHODS FOR DIGITAL COMPUTERS,  
 C     WILEY, NEW YORK/LONDON, 1960, PP.110-120.

C     SOME NOTES ON THE PROGRAM/RALSTON AND WILF

C     AUX

C     ---

C     AUX(1,I) -- CURRENT VALUE OF Y  
 C     AUX(2,I) -- CURRENT VALUE OF Y'  
 C     AUX(3,I) -- LAST GOOD VALUES OF Q  
 C     AUX(4,I) -- Y AFTER ONE RK STEP H  
 C     AUX(5,I) -- Y AFTER ONE OR TWO RK STEPS OF H/2.  
 C     AUX(6,I) -- CURRENT VALUES OF Q  
 C     AUX(7,I) -- Y' AFTER ONE OR TWO RK STEPS OF H/2.  
 C     AUX(8,I) -- 2/15 \* WEIGHTS

C     A(4),B(4),C(4)

C     -----

C     
$$Y_i = Y_{i-1} + A_i * (K_i - B_i * Q_{i-1})$$

C     
$$Q_i = Q_{i-1} + 3 * (A_i * (K_i - B_i * Q_{i-1}) - C_i * K_i)$$

C     FOR VALUES OF I BETWEEN 1 AND 4, AND FOR VALUES OF K AS FOLLOWS

C     
$$K_1 = H * F(X, Y)$$
  
 C     1           0   0

C     
$$K_2 = H * F(X + H/2, Y)$$
  
 C     2           0   1

C     
$$K_3 = H * F(X + H/2, Y)$$
  
 C     3           0   2

C     3 -- sysdesadis:RKGST.FOR

```

C
C      K  = H * F(X+H,Y )
C      4      0      3
C
C      RELATIVE ERROR
C      -----
C
C      AS PER RICHARDSON QUOTED IN RALSTON/WILF (P117),
C
C      ABS ERROR = WEIGHT/15*ABS(Y2 - Y1)
C
C      THEN, RELATIVE ERROR
C
C      REL ERROR = WEIGHT*2/15*ABS(Y2 - Y1)/SUM
C
C      where SUM = ABS(Y2 + Y1)
C
C      The solution tries to use SUM=abs(y2+y1) first. If this is zero,
C      then SUM=.25*ABS(Y1) is used since y1 and y2 must be oposite in
C      sign with equal magnitude. If this
C      quantity is zero as well, the values y2 and y1 both must be zero;
C      therefore, the difference is also zero which satisfies the
C      error criteria.
C
C      .....
C
C      SUBROUTINE RKGST(PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C
C      PARAMETER (EXPAND=0.05)
C      DATA ERRSET/1./      ! dummy value
C
C      DIMENSION Y(1),DERY(1),AUX(8,NDIM),A(4),B(4),C(4),PRMT(1)
C
C      DO 10 I=1,NDIM
10  AUX(8,I)=.13333333*DERY(I)
C      X=PRMT(1)
C      XEND=PRMT(2)
C      H=PRMT(3)
C      IHLFMX = INT(dLOG(ABS(PRMT(5))/PRMT(3)))/.6931472 + .5)
C      PRMT(5)=0.
C      CALL FCT(X,Y,DERY,PRMT)
C
C      C*** ERROR TEST
C
C      IF(H*(XEND-X))380,370,20
C
C      C*** PREPARATIONS FOR RUNGE-KUTTA METHOD
C
C      4 -- sys3desadis:RKGST.FOR

```

```

C
20 A(1)=.5
   A(2)=.2928932
   A(3)=1.707107
   A(4)=.1666667
   B(1)=2.
   B(2)=1.
   B(3)=1.
   B(4)=2.
   C(1)=.5
   C(2)=.2928932
   C(3)=1.707107
   C(4)=.5
C
C*** PREPARATIONS OF FIRST RUNGE-KUTTA STEP
C
   DO 30 I=1,NDIM
   AUX(1,I)=Y(I)
   AUX(2,I)=DERY(I)
   AUX(3,I)=0.
30  AUX(4,I)=0.
   IREC=0
   H=H+H
   INLF=-1
   ISTEP=0
   IEND=0
C
C*** START OF A RUNGE-KUTTA STEP
C*** STEP = 2 * SPECIFIED STEP
C
   40 IF((X+H-XEND)*H)70,60,50
   50 H=XEND-X
   60 IEND=1
C
C*** RECORDING OF INITIAL VALUES OF THIS STEP
C
   70 CALL FCT(X,Y,DERY,PRMT)
   CALL OUTP(X,Y,DERY,IREC,NDIM,PRMT)
   IF (PRMT(5))400,80,400
   80 ITEST=0
   90 ISTEP=ISTEP+1
C
C*** START OF INNERMOST RUNGE-KUTTA LOOP
C
   J=1
100  AJ=A(J)
   BJ=B(J)
   CJ=C(J)
   DO 110 I=1,NDIM
   R1=H*DERY(I)
   R2=AJ*(R1-BJ*AUX(4,I))
3  -- sws$desadis:RKGST.FOR

```

```

      Y(I)=Y(I)+R2
      R2=R2+R2+R2
110  AUX(6,I)=AUX(6,I)+R2-CJ*R1
      IF(J-4)120,150,150
120  J=J+1
      IF(J-3)130,140,130
130  X=X+H/2.
140  CALL FCT(X,Y,DERY,PRMT)
      GOTO 100
C
C*** END OF INNERMOST RUNGE-KUTTA LOOP
C
C*** TEST OF ACCURACY
C
      150 IF(ITEST)160,160,200
C
C*** IN CASE ITEST=0 THERE IS NO POSSIBILITY FOR TESTING OF ACCURACY
C*** IF(ITEST=0) RK STEP JUST PERFORMED WAS FOR TWICE THE SPECIFIED STEP
C
      160 DO 170 I=1,NDIM
      170 AUX(4,I)=Y(I)
          ITEST=1
          ISTEP=ISTEP+ISTEP-2
180  IHLF=IHLF+1
          X=X-H
          H=H/2.
          DO 190 I=1,NDIM
          Y(I)=AUX(1,I)
          DERY(I)=AUX(2,I)
190  AUX(6,I)=AUX(3,I)
          GOTO 90
C
C*** IN CASE ITEST=1 TESTING OF ACCURACY IS POSSIBLE ONLY IF EACH
C*** HALF OF THE INTERVAL IS DONE(I.E.,IFF ISTEP IS EVEN)
C
      200 IMOD=ISTEP/2
          IF(ISTEP-IMOD-IMOD)210,230,210
210  CALL FCT(X,Y,DERY,PRMT)
          DO 220 I=1,NDIM
          AUX(5,I)=Y(I)
220  AUX(7,I)=DERY(I)
          GOTO 90
C
C*** ORIGINAL VERSION: absolute error
C
C      COMPUTATION OF TEST VALUE DELT
C 230      DELT=0.                      !Good so far
C      DO 240 I=1,NDIM
C 240      DELT=DELT+AUX(8,I)*ABS(AUX(4,I)-Y(I))
C      IF(DELT-PRMT(4))290,280,250
C
      4 -- sys$de$adis:RKGST.FOR

```

C\*\*\* RELATIVE ERROR

C

```

230 DELT = 0.
    DO 240 I=1,NDIM
        ARG = ABS(AUX(4,I) + Y(I))
        IF(ARG .EQ. 0.) ARG = .25*ABS(AUX(4,I))
        IF(ARG .EQ. 0.) ARG = ERRSET !if here,aux(4,i)=y(i)=0.0; rer=0.
        RER = AUX(8,I)*ABS(AUX(4,I) - Y(I))/ARG
240 DELT = dMAX1(DELT,RER)
    IF(DELT-PRMT(4)) 280,280,250

```

C

C\*\*\* ERROR IS TOO GREAT

C

```

250 IF(IHLF-10)260,360,360
260 DO 270 I=1,NDIM
270 AUX(4,I)=AUX(5,I)
C    WRITE(5,1200) DELT
C 1200    FORMAT(' ?RKGST? -- ERROR TOO GREAT',613.5)
    ISTEP=ISTEP+ISTEP-4
    X=X-H
    IEND=0
    GOTO 130

```

C

C\*\*\* RESULT VALUES ARE GOOD

C

```

280 CALL FCT(X,Y,DERY,PRMT)
    DO 290 I=1,NDIM
        AUX(1,I)=Y(I)
        AUX(2,I)=DERY(I)
        AUX(3,I)=AUX(6,I)
        Y(I)=AUX(5,I)
290 DERY(I)=AUX(7,I)
    CALL FCT(X-H,Y,DERY,PRMT)
    CALL OUTP(X-H,Y,DERY,IHLF,NDIM,PRMT)
    IF(PRMT(5))400,300,400
300 DO 310 I=1,NDIM
    Y(I)=AUX(1,I)
310 DERY(I)=AUX(2,I)
    IREC=IHLF
    IF(IEND)320,320,390

```

C

C\*\*\* INCREMENT GETS DOUBLED TO KEEP UP WITH ERROR HALVING

C

```

320 IHLF=IHLF-1
    ISTEP=ISTEP/2
    H=H+H

```

C

C\*\*\* ALLOW THE PROGRAM TO EXPAND BEYOND ORIGINAL STEP SIZE SPECIFICATION

C\*\*\* UP TO THE MAXIMUM

C

```

    IF(IHLF+IHLFMX)40,330,330

```

7 -- sys\$desadis:RKGST.FOR

```

330 IMOD=ISTEP/2
    IF(ISTEP-IMOD-IMOD)40,340,40

```

C

```

C*** EXPAND H DUE TO LOW ERROR VALUE
C*** ONLY IF TWO CONSECUTIVE STEPS WITHOUT MENTION OF ERROR HAVE
C*** COMPLETED. FACTOR USED FOR EXPANSION (EXPAND) WAS SHOWN TO WORK
C*** REASONABLY WELL FOR A PERIODIC FUNCTION. VALUES AS HIGH AS
C*** EXPAND=.5 WILL PRODUCE GOOD RESULTS FOR MONOTONIC FUNCTIONS.

```

C

```

340 IF(DELTA-EXPAND*PRMT(4)) 350,350,40
350 IHLF=IHLF-1
    ISTEP=ISTEP/2
    H=H+H
    GOTO 40

```

C

```

C*** RETURNS TO CALLING PROGRAM

```

C

```

360 IHLF=11
    CALL FCT(X,Y,DERY,PRMT)
    GOTO 390
370 IHLF=12
    GOTO 390
380 IHLF=13
390 CALL FCT(X,Y,DERY,PRMT)
    CALL OUTP(X,Y,DERY,IHLF,NDIN,PRMT)
400 RETURN
    END

```

####

.....

SUBROUTINE RTMI

.....

This routine was originally supplied by Digital Equipment Corporation as part of the Scientific Subroutine Package available for RT-11 as part of the Fortran Enhancement Package. It was adopted for use in this package.

.....

## PURPOSE

TO SOLVE GENERAL NONLINEAR EQUATIONS OF THE FORM  $FCT(X)=0$   
BY MEANS OF MUELLER-S ITERATION METHOD.

## USAGE

CALL RTMI (X,F,FCT,XLI,XRI,EPS,IEND,IER)  
PARAMETER FCT REQUIRES AN EXTERNAL STATEMENT.

## DESCRIPTION OF PARAMETERS

X - RESULTANT ROOT OF EQUATION  $FCT(X)=0$ .  
F - RESULTANT FUNCTION VALUE AT ROOT X.  
FCT - NAME OF THE EXTERNAL FUNCTION SUBPROGRAM USED.  
XLI - INPUT VALUE WHICH SPECIFIES THE INITIAL LEFT BOUND  
OF THE ROOT X.  
XRI - INPUT VALUE WHICH SPECIFIES THE INITIAL RIGHT BOUND  
OF THE ROOT X.  
EPS - INPUT VALUE WHICH SPECIFIES THE UPPER BOUND OF THE  
ERROR OF RESULT X.  
IEND - MAXIMUM NUMBER OF ITERATION STEPS SPECIFIED.  
IER - RESULTANT ERROR PARAMETER CODED AS FOLLOWS  
IER=0 - NO ERROR,  
IER=1 - NO CONVERGENCE AFTER IEND ITERATION STEPS  
FOLLOWED BY IEND SUCCESSIVE STEPS OF  
BISECTION,  
IER=2 - BASIC ASSUMPTION  $FCT(XLI)*FCT(XRI)$  LESS  
THAN OR EQUAL TO ZERO IS NOT SATISFIED.

## REMARKS

THE PROCEDURE ASSUMES THAT FUNCTION VALUES AT INITIAL  
BOUNDS XLI AND XRI HAVE NOT THE SAME SIGN. IF THIS BASIC  
ASSUMPTION IS NOT SATISFIED BY INPUT VALUES XLI AND XRI, THE  
PROCEDURE IS BYPASSED AND GIVES THE ERROR MESSAGE IER=2.

## SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED

THE EXTERNAL FUNCTION SUBPROGRAM FCT(X) MUST BE FURNISHED  
BY THE USER.

```

C      METHOD
C      SOLUTION OF EQUATION FCT(X)=0 IS DONE BY MEANS OF MUELLER-S
C      ITERATION METHOD OF SUCCESSIVE BISECTIONS AND INVERSE
C      PARABOLIC INTERPOLATION, WHICH STARTS AT THE INITIAL BOUNDS
C      XLI AND XRI. CONVERGENCE IS QUADRATIC IF THE DERIVATIVE OF
C      FCT(X) AT ROOT X IS NOT EQUAL TO ZERO. ONE ITERATION STEP
C      REQUIRES TWO EVALUATIONS OF FCT(X). FOR TEST ON SATISFACTORY
C      ACCURACY SEE FORMULAE (3,4) OF MATHEMATICAL DESCRIPTION.
C      FOR REFERENCE, SEE G. K. KRISTIANSEN, ZERO OF ARBITRARY
C      FUNCTION, BIT, VOL. 3 (1963), PP.205-206.
C
C      .....
C
C      SUBROUTINE RTMI(X,F,FCT,XLI,XRI,EPS,IEND,IER)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      PREPARE ITERATION
C      IER=0
C      XL=XLI
C      XR=XRI
C      X=XL
C      TOL=X
C      F=FCT(TOL)
C      IF(F)1,16,1
C 1  FL=F
C      X=XR
C      TOL=X
C      F=FCT(TOL)
C      IF(F)2,16,2
C 2  FR=F
C      IF(dSIGN(1.D0,FL)+dSIGN(1.D0,FR))25,3,25
C
C      BASIC ASSUMPTION FL*FR LESS THAN 0 IS SATISFIED.
C      GENERATE TOLERANCE FOR FUNCTION VALUES.
C 3  I=0
C      TOLF=100.*EPS
C
C      START ITERATION LOOP
C 4  I=I+1
C
C      START BISECTION LOOP
C      DO 13 K=1,IEND
C      X=.5*(XL+XR)
C      TOL=X
C      F=FCT(TOL)
C      IF(F)5,16,5
C 5  IF(dSIGN(1.D0,F)+dSIGN(1.D0,FR))7,6,7
C
C 2 -- see degadis:RTMI.FOR

```



```

C
C   INTERCHANGE XL AND XR IN ORDER TO GET THE SAME SIGN IN F AND FR
6  TOL=XL
   XL=XR
   XR=TOL
   TOL=FL
   FL=FR
   FR=TOL
7  TOL=F-FL
   A=F*TOL
   A=A+A
   IF(A-FR*(FR-FL))8,9,9
9  IF(I-IEND)17,17,9
   XR=X
   FR=F

C
C   TEST ON SATISFACTORY ACCURACY IN BISECTION LOOP
   TOL=EPS
   A=ABS(XR)
   IF(A-1.)11,11,10
10  TOL=TOL*A
11  IF(DABS(XR-XL)-TOL)12,12,13
12  IF(DABS(FR-FL)-TOLF)14,14,13
13  CONTINUE
C   END OF BISECTION LOOP
C
C   NO CONVERGENCE AFTER IEND ITERATION STEPS FOLLOWED BY IEND
C   SUCCESSIVE STEPS OF BISECTION OR STEADILY INCREASING FUNCTION
C   VALUES AT RIGHT BOUNDS. ERROR RETURN.
   IER=1
14  IF(DABS(FR)-DABS(FL))16,16,15
15  X=XL
   F=FL
16  RETURN

C
C   COMPUTATION OF ITERATED X-VALUE BY INVERSE PARABOLIC INTERPOLATION
17  A=FR-F
   DX=(X-XL)*FL*(1.+F*(A-TOL)/(A*(FR-FL)))/TOL
   XM=X
   FM=F
   X=XL-DX
   TOL=X
   F=FCT(TOL)
   IF(F)18,16,18

C
C   TEST ON SATISFACTORY ACCURACY IN ITERATION LOOP
18  TOL=EPS
   A=ABS(X)
   IF(A-1.)20,20,19
19  TOL=TOL*A
20  IF(DABS(DX)-TOL)21,21,22

3  -- sussdesadis:RTMI.FOR

```

```

21 IF(dABS(F)-TOLF)16,16,22
C
C   PREPARATION OF NEXT BISECTION LOOP
22 IF(DSIGN(1.D0,F)+dSIGN(1.D0,FL))24,23,24
23 XR=X
   FR=F
   GO TO 4
24 XL=X
   FL=F
   XR=XM
   FR=FM
   GO TO 4
C
C   END OF ITERATION LOOP
C
C
C   ERROR RETURN IN CASE OF WRONG INPUT DATA
25 IER=2
   RETURN
   END
****

```

## PROGRAM SDEGADIS2

C  
 C\*\*\*\*\*  
 C\*\*\*\*\*  
 C\*\*\*\*\*

C  
 C Program description:

C  
 C SDEGADIS2 is a simplification of DEGADIS2 which performs the downwind  
 C dispersion portion of the calculation for a steady state source  
 C described by DEGADIS1.

C  
 C Program usage:

C  
 C Consult Volume III of the Final Report to U. S. Coast Guard  
 C contract DT-CG-23-80-C-20029 entitled 'Development of an  
 C Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures'.

C  
 C J. A. Havens  
 C T. O. Spicer

C  
 C University of Arkansas  
 C 227 Engineering Building  
 C Department of Chemical Engineering  
 C Fayetteville, AR 72701

C  
 C April 1985

C  
 C This project was sponsored by the U. S. Coast Guard and the Gas  
 C Research Institute under contract DT-CG-23-80-C-20029.

C  
 C Disclaimer:

C  
 C This computer code material was prepared by the University of  
 C Arkansas as an account of work sponsored by the U. S. Coast Guard  
 C and the Gas Research Institute. Neither the University of Arkansas,  
 C nor any person acting on its behalf:

- C  
 C a. Makes any warranty or representation, express or implied,  
 C with respect to the accuracy, completeness, or usefulness  
 C of the information contained in this computer code material,  
 C or that the use of any apparatus, method, numerical model,  
 C or process disclosed in this computer code material may not  
 C infringe privately owned rights; or
- C  
 C b. Assumes any liability with respect to the use of, or for  
 C damages resulting from the use of, any information,  
 C apparatus, method, or process disclosed in this computer

1 -- sys\$desadis:SDEGADIS2.FOR

```

C      code material.
C
C
C

```

```

C*****
C*****
C*****
C

```

```

C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

```

```

C      include 'sys$desadis:DEGADIS2.dec'
C      include '($ssdef)'

```

```

C      EXTERNAL PSS,PSSOUT,SSG,SSGOUT

```

```

C      COMMON

```

```

C      $/TITL/ TITLE
C      $/GEN2/ DEN(5,iden)
C      $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/COM_EPRON/ sas_mw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C      $ sas_ufl,sas_lfl,sas_zsp,sas_name
C      $/ITI/ TI,TINP,TSRC,TOBS
C      $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wtco
C      $/ERROR/ SYOER,ERRP,SMXP,WTSZP,WTSYP,WTBEP, WTDH,ERRG,SMXG,
C      $ WTRUH,WTDHG
C      $/STP/ STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C      $/com_ss/ ESS,SLEN,SWID,OUTC,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
C      $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      $/com_fl/ cflas,cfl,cufl
C      $/NEND/ POUNDN,POUND
C      $/ALP/ ALPHA,alpha1
C      $/phicom/ iphifl,dellay
C      $/sprd_ccn/ ce, delrhomin
C      $/COM_SURF/ HTCUT

```

```

C      DIMENSIONS/DECLARATIONS

```

```

C      logical cflas

```

```

C      real*4 tt1
C      REAL*8 K,ML,L
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5

```

```

C      character*24 tinp,tsrc,tobs
C      character*80 title(4)

```

```

C      -- sys$desadis:SDEGADIS2.FOR

```

```

character*4 pound
character*3 sas_name

C
character*4 TR2,ER2,Sr3,SSD,TR3
C
character*40 opnrup1
character opnrup(40)
EQUIVALENCE (OPNRUP(1),opnrup1)
C
dimension prnt(22),v(4),dery(4),aux(8,4)
C
C.....
C
C DATA
C
C DATA POUND/'/'/'/'/'',POUNDN/-1.E-20/
C
C DATA TIME0/0./,NDIM/0/
C
C DATA TR2/''.TR2''/,ER2/''.ER2''/
C DATA Sr3/''.Sr3''/
C DATA SSD/''.SSD''/,TR3/''.TR3''/
C
C.....
C
C MAIN
C
C T1 = SECNDS(0.)
C istat = lib$date_time(TOBS)
C if(istat .ne. ss$normal) stop 'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
C read(5,1135) nchar,opnrup
C 1135 format(a,40a1)
C
C opnrup1 = opnrup1(1:nchar) // ER2(1:4)
C CALL ESTR2ss(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
C opnrup1 = opnrup1(1:nchar) // tr2(1:4)
C CALL STRT2(OPNRUP1,H..a:srte)
C
C opnrup1 = opnrup1(1:nchar) // sr3(1:4)
C OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP1,
C 8 CARRIAGECONTROL='FORTRAN')
C
C cflag = isof1.eq. 1.or. ihtf1.eq. 0
C
3 -- sys$desadis:SDEGADIS2.FOR

```

```

WRITE(8,1119)
if(cflas) then
    WRITE(8,1116) (100.*sas_lfl),(100.*sas_ufl),sas_zsp
    WRITE(8,1118)
else
    WRITE(8,1115) (100.*sas_lfl),(100.*sas_ufl),sas_zsp
    WRITE(8,1117)
endif
WRITE(8,1119)

C
C
C
1115 FORMAT(1H0,1X,'Distance',2X,3X,'Mole',3X,
1      'Concentration',1X,'Density',2X,3X,'Gamma',4X,
1      'Temperature',3X,'Half',4X,4X,'Sz',5X,4X,'Sw',5X,
1      1X,'Width to',3X,'Width',/,1X,11X,1X,'Fraction',2X,
1      11X,11X,11X,11X,3X,'Width',3X,11X,9X,
1      2(1P9.3,'mole%',1X),/,1h ,
1      99X,4X,'at z= ',0P6.2,' m')
1116 FORMAT(1H0,1X,'Distance',2X,3X,'Mole',3X,
1      'Concentration',1X,'Density',2X,
1      'Temperature',3X,'Half',4X,4X,'Sz',5X,4X,'Sw',5X,
1      1X,'Width to',3X,'Width',/,1X,11X,1X,'Fraction',2X,
1      11X,11X,11X,3X,'Width',3X,11X,9X,
1      2(1P9.3,'mole%',1X),/,1h ,
1      99X,4X,'at z= ',0P6.2,' m')
1117 FORMAT(1H ,4X,'(m)',4X,11X,
1      2(1X,'(kg/m**3)',1X),12X,4X,'(K)',
1      5(8X,'(m)'))
1118 FORMAT(1H ,4X,'(m)',4X,11X,
1      2(1X,'(kg/m**3)',1X),4X,'(K)',
1      5(8X,'(m)'))
1119 FORMAT(1H )

C
C
C
C.....
C
C      STEADY STATE CALCULATIONS
C
C
C      opnrup1 = opnrup1(1:nchar) // ssd(1:4)
C      OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1)
C
C
C      AGAIN = .FALSE.
C
C
C      L = OUTL
C      B = OUTB
C      SZO = OUTSZ

4 -- sys$desadis:SDEGADIS2.FOR

```

```

      Erate = ESS
      QSTRO = Erate/2./L/B
      Cc = OUTCc
      wc = swcl
      wa = swal
      enth = senl
      rho = srhl
C
      call setden(wc,wa,enth)
C
200  if(cf1as) then
          call adiabab(2,twc,twa,gas_lfl,ga,cfl,r,w,t,tt)
          call adiabab(2,twc,twa,gas_ufl,ga,cufl,r,w,t,tt)
          endif
      ratio1= u0*z0/alpha1/ z0**alpha1 * cc /b/estr0/1
      ratio = ratio1* sz0**alpha1 * (B + sarrpi/2.*sy0er)
      if(ratio.le. 1.) then
          sy0er = (1./((ratio1*sz0**alpha1) - b)*2./sarrpi
      else
          sz0 = (1./((B+ sarrpi/2.*sy0er)*ratio1))**(1./alpha1)
      endif
      if(cc.gt. rhoe) then
          write(lunlos,1126) cc,rhoe
1126      format(/,' ',10('***'),/, ' cc: ',1ps13.5,' is greater',
1          ' than rhoe: ',1ps13.5/, ' ',10('***'),/)
          cc =rhoe
          endif
C
      cclay = cc/dellay
      call adiabab(0,wclay,walay,wclay,walay,cclay,rholay,w,enthlay,t)
C
C
C*** let everyone know
C
      WRITE(lunlos,1170) L,B
      WRITE(lunlos,1180) QSTRO,SZ0
      write(lunlos,1185) wclay,walay,rholay,cclay,t
      write(lunlos,1186) wc,wa,rho,cc,temp
C
1170  FORMAT(' LENGTH: ',1ps13.5,' BEFF: ',1ps13.5)
1180  FORMAT(' TAKEUP FLUX: ',1ps13.5,' SZ0: ',1ps13.5)
1185  format(' wclay: ',1ps12.5,' walay: ',1ps12.5,
1      ' rholay: ',1ps12.5,' Cclay: ',1ps12.5/,
1      ' temlay: ',1ps13.5)
1186  format(' wc: ',1ps12.5,' wa: ',1ps13.5/,
1      ' rho: ',1ps12.5,' Cc: ',1ps12.5,' temp: ',1ps12.5)
C
C*** PREPARE FOR STEADY STATE INTEGRATION.
C
      PRMT(1) = L/2.
      PRMT(2) = 6.023E13

5 -- sysidesadis:SDEGADIS2.FOR

```

```

      PRMT(3) = STPP
      PRMT(4) = ERRP
      PRMT(5) = SMXP
      PRMT(6) = Erate
      PRMT(7) = Cc      ! OUTPUT
      PRMT(8) = B       ! OUTPUT
C
C*** PRMT(9) & PRMT(10) ARE CONSTANTS FOR D(SY) & D(SZ)
C
      PRMT(9) = Ce*sqrt(G*Z0/ALPHA1*GAMMAF) *GAMMAF/U0
      PRMT(10)= Z0**ALPHA*K*USTAR*ALPHA1 * ALPHA1/U0
C
      PRMT(11)= NREC
C
      PRMT(12)=
C
      PRMT(13)=
      prmt(18)= u0*z0/alpha1
      prmt(19)= rhoa*k*ustar*alpha1
      prmt(20)= rho1ay
      prmt(21)= sz0
      prmt(22)= sz0
C
      Y(1) = rho1ay*prmt(18)*(SZ0/z0)**alpha1 ! rho1ay*ueff*heff
      Y(2) = SYOER
      Y(3) = B + sqrt(pi/2.*sy0er
      Y(4) = 0.                                ! added heat
C
      DERY(1) = WTSZP
      DERY(2) = WTSYP
      DERY(3) = WTBEP
      dery(4) = wtdh
C
      NDIM = 4
C
      WRITE(lunlog,1130)
1130 FORMAT(' Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
      NREC = INT(PRMT(11))
      WRITE(lunlog,1100)NREC
1100 FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10)
C
      IF(AGAIN) GO TO 120
C
C*** GAUSSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C*** FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
6 -- sys$desadis:SDEGADIS2.FOR

```



```

C*** THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C*** CALCULATED AS IF SEFF=SY RETAINING THE LAST VALUE OF Cc IN THE
C*** MATERIAL BALANCE.
C
      heat = Y(4)
      rho1ay = PRMT(20)
      Cc = PRMT(7)
      rho1uh = Y(1)
      SZ = ( rho1uh/rho1ay/PRMT(18) )**(1./ALPHA1) * Z0
      SYT = Erate*ALPHA1 *(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
      XT = PRMT(12)
      XV = (SYT/RT2/DELTA)**(1./BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
      do i=1,22
        PRMT(i) = 0.
      enddo
C
      PRMT(1) = XT
      PRMT(2) = 6.023E23
      PRMT(3) = STPG
      PRMT(4) = ERG
      PRMT(5) = SMXG
      PRMT(6) = Erate
      PRMT(7) = Cc          !-- OUTPUT
      PRMT(8) = XV
C
      PRMT(9) = 'BLANK'
C
      PRMT(10) = 'BLANK'
C
      PRMT(11) = 'BLANK'
C
      PRMT(12) = DIST AT COMPLETION -- OUTPUT
C
      PRMT(18) = U0*Z0/ALPHA1
      PRMT(19) = rho1uh/rho1ay/ALPHA1
      PRMT(20) = rho1ay
      PRMT(21) = SZ
      PRMT(22) = SZ
C
      Y(1) = rho1uh
      Y(2) = heat
C
      DERY(1) = WTRUH
      DERY(2) = WTDHS
C
      NDIM = 2
C
      WRITE(1140,1140)
      1140 FORMAT(' Entering Gaussian Stage of Integration ')
7 -- sys$desadis:SDEGADIS2.FOR

```

```

C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(10,IHLF)
C
      NREC = INT(PRMT(11))
C
120  CONTINUE
C
C
C.....
C
      CLOSE(UNIT=9)
      CLOSE(UNIT=8)
C
      opnrup1 = opnrup1(1:nchar) // tr3(1:4)
      CALL TRANS(OPNRUP1)
C
      tt1 = t1
      T1 = SECNDS(tt1)/60.
      WRITE(lunlos,4000) TOBS
      WRITE(lunlos,4010) T1
4000 FORMAT(//,'SDEGADIS2 -->',//,3X,'BEGAN AT ',A40)
4010 FORMAT(3X,'*** ELAPSED TIME *** ',1P13.5,' min')
C
      STOP
      END
****

```

```

C.....
C
C   TIME SORT SUPERVISOR
C
C   SUBROUTINE SORTS(TABLE)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
C   include 'sys$desadis:DEGADIS3.dec/list'
C
C   COMMON
C   $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
C   $       Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
C   $       Tssaaa(maxnob,maxnt),Ttemp(maxnob,maxnt),
C   $       TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
C   $       TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
C   $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
C   $/SORTIN/ TIM(maxnt),NTIM,ISTR
C   $/PARAM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/CNOBS/ NOBS

C   DIMENSION TABLE(1)

C   REAL*8 ML,K

C   CALL GETTIM

C
C
C*** TABLE(I) VALUES
C*** I          PARAMETER
C*** --
C*** 1 11      DIST          1 TO 10 CURRENT READ
C*** 2 12      Yc
C*** 3 13      Cc          11 TO 20 PREVIOUS READ
C*** 4 14      rho
C*** 5 15      sssaaa
C*** 6 16      temp
C*** 7 17      SZ
C*** 8 18      SY
C*** 9 19      B
C*** 10 20     TS
C
C*** 21      DISTO
C*** 22      INTERPOLATION FRACTION
C
C   DO 100 I = 1,NOBS
C
C       IT = 0
C
C   DO 105 J=1,20

1 -- sys$desadis:SORTS.FOR

```

```

105 TABLE(J) = 0.
C
    II = NREC(I,1)

    if( ii .eq. 0) goto 130
C
c*** read first record
C
    read(9,*) (table(k1),k1=1,9)
    table(10) = ts( t0(I), table(1) )
    table(21) = table(1)
C
c*** loop through and read each record even if not pertinent
C
    DO 110 J = 2,II
C
        DO K1 = 1,10
        KK = K1 + 10
        TABLE(KK) = TABLE(K1)
        enddo
C
        READ(9,*) (TABLE(K1),K1=1,9)
        TABLE(10) = TS( T0(I), TABLE(1) )
C
        itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
        itl = min( ntim, itl)
        itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
        itf = max( 1, itf)

        do it = itf, itl, 1                ! do all points in range
C
c*** RECORD AN INTERPOLATED TIME SORTED POINT.
C
        KSUB(IT) = KSUB(IT) + 1
C
        TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
        TDISTO(I,IT) = TABLE(21)
        TDIST(I,IT) = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
        Tvc(I,IT) = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
        TCc(I,IT) = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
        Trho(I,IT) = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
        Tgamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
        Ttemp(I,IT) = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
        TSZ(I,IT) = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
        TSY(I,IT) = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
        TB(I,IT) = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
        enddo
110 CONTINUE
C
2 -- sys$desadis:SORTS.FOR

```

```

C
130 II = NREC(I,2)
    IF(II .EQ. 0) GO TO 100

    DO 200 J=1,II
C
        DO K1 = 1,10
            KK = K1 + 10
            TABLE(KK) = TABLE(K1)
        enddo
C
        READ(9,*) (TABLE(K1),K1=1,7)
C
        TABLE(8) = RT2*DELTA*(TABLE(1) + XV(I))*BETA
        TABLE(9) = 0.
        TABLE(10) = TS(TO(I),TABLE(1))
C
        itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
        itl = min( ntim, itl)
        itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
        itf = max( 1, itf)

        do it = itf, itl, 1                ! do all points in range
C
C*** RECORD A TIME SORTED VALUE
C
        KSUB(IT) = KSUB(IT) + 1
        TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
        TDISTO(I,IT) = TABLE(21)
        TDIST(I,IT) = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
        Tyc(I,IT) = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
        Tcc(I,IT) = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
        Trho(I,IT) = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
        Tsamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
        Ttemp(I,IT) = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
        TSZ(I,IT) = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
        TSY(I,IT) = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
        TB(I,IT) = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
        enddo
    200 CONTINUE
C
    100 CONTINUE
C
    CALL SORTS1(TABLE)
C
    RETURN
    END
****

```

```

C.....
C
C
C      SUBROUTINE SORTS1(TABLE)

C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
C      include 'sys$desadis:DEGADIS3.dec/list'
C
C      COMMON
C      $/SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
C      $      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
C      $      Tsamma(maxnob,maxnt),Tteap(maxnob,maxnt),
C      $      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
C      $      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
C      $/SSCON/NREC(maxnob,2),T0(maxnob),XV(maxnob)
C      $/SORTIN/TIM(maxnt),NTIM,ISTR1
C      $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iutfl,wteo
C      $/PARMSC/RH,SZH,EMAX,RMAX,TSC1,ALEPH,TEND
C      $/ccm_six/ six_coeff,six_pow,six_min_dist,six_flg
C      $/CNOBS/NOBS
C
C      DIMENSION TABLE(1)
C
C      REAL*8 ML,K
C
C      C*** DETERMINE IF ANY TIME VECTORS HAVE NO ENTRIES
C
C      DO 192 I=1,NTIM
C      192 IF(KSUB(I).GT. 2) GO TO 194      ! 1or2 points is of little value
C      call trap(23)
C      194 ISTR1 = I
C      DO 196 I=ISTR1,NTIM
C      196 IF(KSUB(I).LE. 2) GO TO 198
C      GO TO 199
C      198 NTIM = I - 1
C      199 CONTINUE
C
C      C*** REVERSE TIME SORTED VECTORS
C
C      DO 200 K1 = ISTR1,NTIM
C
C      II = KSUB(K1)
C      DO 170 J = 1,NOBS
C
C      IF(TDIST(J,K1).NE.0. .OR. TB(J,K1).NE.0.) GO TO 180
C      170 CONTINUE
C      180 II = II + J - 1
C
C      DO 190 J = 1,II

```

! -- sys\$desadis:SORTS1.FOR

```

C
TABLE(II +      + 1 - J) = TCc(J,K1)
TABLE(II + NOBS + 1 - J) = Tvc(J,K1)
TABLE(II + 2*NOBS + 1 - J) = Trho(J,K1)
TABLE(II + 3*NOBS + 1 - J) = Tsgama(J,K1)
TABLE(II + 4*NOBS + 1 - J) = Ttemp(J,K1)
TABLE(II + 5*NOBS + 1 - J) = TSY(J,K1)
TABLE(II + 6*NOBS + 1 - J) = TSZ(J,K1)
TABLE(II + 7*NOBS + 1 - J) = TB(J,K1)
TABLE(II + 8*NOBS + 1 - J) = TDISTO(J,K1)
TABLE(II + 9*NOBS + 1 - J) = TDIST(J,K1)

C
190 CONTINUE
C
DO 210 J = 1,II
C
TCc(J,K1) = TABLE(J)
Tvc(J,K1) = TABLE(J + NOBS)
Trho(J,K1) = TABLE(J + 2*NOBS)
Tsgama(J,K1) = TABLE(J + 3*NOBS)
Ttemp(J,K1) = TABLE(J + 4*NOBS)
TSY(J,K1) = TABLE(J + 5*NOBS)
TSZ(J,K1) = TABLE(J + 6*NOBS)
TB(J,K1) = TABLE(J + 7*NOBS)
TDISTO(J,K1) = TABLE(J + 8*NOBS)
TDIST(J,K1) = TABLE(J + 9*NOBS)

C
210 CONTINUE
200 CONTINUE

C
if(six_flg.eq. 0.) then                ! no correction
C
write(lunlog,*) ' No X-direction dispersion correction'
DO 220 K1 = ISTRT,NTIM
II = KSUB(K1)
DO 220 I = 1,II
TCcSTR(I,K1) = TCc(i,k1)
220 CONTINUE
return
endif

C
C*** GENERATE TCcSTR -- CENTER LINE CONCENTRATION CORRECTED FOR
C*** DOWNWIND DISPERSION.
C
DO 230 K1 = ISTRT,NTIM
C
II = KSUB(K1)
DO 240 I = 1,II
C
c calculation for XP = TDIST(I,K1)

2 -- sys$desadis:SORTS1.FOR

```

```

C      TCcSTR(I,K1) = 0.
C
C      DO 260 J = 1,II
C
C          TABLE(J) = 0.
C          DIST = TDIST(J,K1) + RMAX
C          DIST = TDIST(J,K1) - Tdist0(J,K1)
C          delta = ABS(tdist(i,k1) - tdist(J,k1))
C
C          if(dist.lt. sisx_min_dist) then
C              if(i.eq. J) then ! i.e. delta = 0.
C                  table(J) = (tdist(J+1,k1)- tdist(J-1,k1))/2.
C                  if(J.eq.1)table(J)= (tdist(2,k1)- tdist(1,k1))/2.
C                  if(J.eq. ii)table(J)=tdist(ii,k1)- tdist(ii-1,k1)
C                  table(J) = TCc(J,k1) / table(J) * RT2/SQRTPI
C              endif
C              goto 260
C          endif
C
C          SX = sisx_coeff* DIST**sisx_pow
C
C          ARG = (delta/SX)**2/2.
C
C          IF(ARG .EQ. 0.) TABLE(J) = TCc(J,K1)/SX
C          IF(ARG .NE. 0. .AND. ARG .LE. 90.)
C              $      TABLE(J) = TCc(J,K1)/SX/EXP(ARG)
C
C      260  CONTINUE
C
C      III = KSUB(K1)
C      TCcSTR(I,K1) = TABLE(1)* (TDIST(2,K1)- TDIST(1,K1))/2.
C      TCcSTR(I,K1) = TABLE(III)* (TDIST(III,K1)- TDIST(III-1,K1))
C      1      + TCcSTR(I,K1)
C      III = ksub(k1) - 1
C
C      DO 280 J = 2,III
C
C          TCcSTR(I,K1) = TABLE(J)* (TDIST(J+1,K1)- TDIST(J-1,K1))/2.
C      1      + TCcSTR(I,K1)
C
C      280  CONTINUE
C
C      TCcSTR(I,K1) = TCcSTR(I,K1)/RT2/SQRTPI
C
C      *** correct uc, rho, and temp values
C
C          cc = Tccstr(i,k1)
C          if(isofl.eq. 1 .or. ihtfl.eq. 0) then
C              call adiabat(0,uc,wa,yc,ys,cc,rho,wm,enth,temp)
C          else
C
C      3 -- sysdesadis:SORTS1.FOR

```



C-110

```
      enth = Tgama(i,K1)
      call adiab(-1,wc,wa,yc,ya,cc,rho,wa,enth,temp)
    endif
    Tyc(i,K1) = yc
    Trho(i,K1)= rho
C
240  CONTINUE
C
230  CONTINUE
C
      RETURN
      END
****
```

C SOURCE EQUATIONS -- Gas Blanket present

SUBROUTINE SRC1(time,Y,D,PRMT)

Implicit Real\*8 ( A-H, O-Z ), Integer\*4 ( I-N )

include 'sys\$desadis:DEGADIS1.dec'

```
parameter(      delt= 0.1,
1             delto2= delt/2.,
2             zero= 1.e-20,
3             rcrit= 0.002)
```

COMMON

```
$/GEN1/ ET(2,isen),R1T(2,isen)
$/ERROR/STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtyc,wteb,wtmb,wtUH,XLI,
$ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srccs,srccut,
$ htcut,ERNOBL,NOBLt,crfser,epsilon
$/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
$/comata/ istab,tamb,paab,humid,isoft,tsurf,ihtfl,htco,iwtfl,wtco
$/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
$/vucom/ vua,vub,vue,vud,vudelta,vufas
$/com_enthal/ h_masrte,h_airrte,h_watrte
$/ALP/ ALPHA,alpha1
$/phicom/ iphifl,dellay
$/sard_con/ ce, delrhomin
```

```
LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
logical vufas
```

```
REAL*8 ML,K
REAL*8 L,masrte,mole
INTEGER R,mass,massc,massa,ebal,abal
DIMENSION Y(7),D(7),PRMT(25)
DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/,abal/6/
```

```
if( prmt(20).lt. 0.D0) vufas = .false.
```

```
if(Y(mass) .le. 0.D0) then
    wc = dmax1(prmt(15),1.d-10)
    if(wc.gt. 1.) wc=1.d-10
    wa = 1. - wc
    enthalpy = wc*h_masrte          ! air contributes nothing
else
    wc = Y(massc)/Y(mass)
    wa = Y(massa)/Y(mass)
    enthalpy = Y(ebal)/Y(mass)
endif
```

1 -- sys\$desadis:SRC1.FOR

```

call tprop(1,uc,wa,enthalpy,vc,va,mole,temp,rho,cp)

RADP = AFGEN(R1T,TIME,'R1TSRC')
hei = Y(mass)/pi/Y(r)/Y(r)/rho
hei = dmax1( Y(mass)/pi/Y(r)/Y(r)/rho , 0.D0 )
delrho = rho-rhoa
dprime = s*delrho/rhoa *hei

C*** CALCULATE D(R),airrte,vel

D(R) = 0.
vel = 0.
airrte = 0.
Ri = 0.
D(mbal) = 0.

IF(Gprime.GT. 0.) then
  slump = Ce*sart(Gprime)

  if(vufas) then                ! momentum balance
    iii = 0                      ! initialize loop counter
    vel = prmt(14)               ! old velocity value
    velmin = 0.
    velmax = dmax1( slump, 0.1D0, vel)

100    hh = vel*vel/Ce/Ce/s/ (delrho/rhoa)
        rh = Y(r)-vua*vub*hh
        value = Y(r)**2/rh**2

        if(prmt(25).se. prmt(24)) then      ! hh .se. ht
          ht = 2.*(value*hei - vua*hh*(value-1.)) - hh
          velc = Y(mbal)/(0.4*pi*rho*(2./3.*ht + hh)*rh**3/Y(r)
1          + 2./3.*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
2          + vuc*pi*Y(r)*hei**2*rhoa)
          D(mbal) = pi*s*delrho*Y(r)*ht**2
1          - vua*vud*pi*rhoa*Y(r)*hh*vel**2
          else
            ht = value*hei - vua*hh*(value-1.)
            velc = Y(mbal)/(2./3.*pi*rho*ht*rh**3/Y(r)
1            + 2./3.*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
2            + vuc*pi*Y(r)*hei**2*rhoa)
            D(mbal) = pi*s*delrho*(rh*ht**2 + vua*vub*hh*hh*hh)
1            - vua*vud*pi*rhoa*Y(r)*hh*vel**2
            endif

          dif = abs(vel-velc)                ! convergence check
          sum = abs(vel) + abs(velc) + zero

          if(dif/sum .le. rcrit) then

2 -- sysfdesadis:SRC1.FOR

```

```

      vel = (vel+velc)/2.
      prmt(13) = vel

      if(vel .gt. 0.) then
        Ri = sprime / vel**2
        airrte= 2.*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
        D(r) = vel
        prmt(20) = slump
      endif
    else
      dif = vel-velc

      if(velc.lt.velain) velain= dmini(velc, 0.D0)
      if(velc.gt.velaax) velaax=velc

      if(dif .gt. 0.) then
        velaax = vel
        vel = 0.5D0*(velaax-velain) + velain
      else
        velain = velc
        vel = (1.D0-0.5D0)*(velaax-velain) + velain
      endif

      iii = iii+1
      if(iii .gt. 40) stop 'SRC1 velocity loop'
      goto 100
    endif

  else
    vel= slump      ! gravity slumping
    hh = hei
    ht = hei
    Ri = sprime / vel**2
    airrte= 2.*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
    D(r) = vel
  endif
endif

C
IF(delrho.Lt.delrhomin .and. .not.(check2 .or. u0.eq.0.))
1   D(R) = 0. ! not for HSE tyres or no wind cases

C
area = pi * (Y(r)**2 - rada**2)
IF(Y(R).LT. RADP) THEN
  AREA = 0.
  Y(R) = RADP
  IF(time.gt. delto2) then ! delti num prob
    D(R) = dmax1(0.D0,
1      ((AFGEN(R1T,TIME+delto2,'R1TSRD')-
3 -- sys$desadis:SRC1.FOR

```

```

2          AFGEN(R1T, (TIME-delta2), 'R1TSRe'))/ delt))
      else
          D(R) = dmax1(0.D0,
1              ((AFGEN(R1T,delta2,'R1TSRD')-
2              AFGEN(R1T, 0., 'R1TSRe'))/ delta2))
      endif
ENDIF

c
c calculate totrteout
c
masrte = AFGEN(ET,TIME,'src1')
L = SQRTPI * Y(R)

c
cc = wc*rho

c
c
astrmx = 0.
if(u0 .ne. 0.)
1  astrmx = cc*K*USTAR*ALPHA1*dellay/(dellay-1.)/phihat(rho,L)

c
c
c
astrll = astrmx * L*L
totrteout = astrll/wc

c
c surface effects
c
watrte = 0.
surface_a = 0.
call surface(temp,hei,rho,mole,cp,watrte,surface_a)
surface_a = area * surface_a
if(surface_a.lt. 0.) surface_a = 0. ! don't let the cloud cool
watrte = area * watrte

c
500 totrtein = airrte + masrte + watrte
c
IF(totrtein.lt.totrteout .and. .not.check2
1    ) then ! check2 is True for HSE type spills
    D(R) = 0.
    totrtein = airrte + masrte + watrte
endif

c
c CALCULATE D(mass),D(massc),D(massa),D(anythings left)
c
D(mass) = totrtein - totrteout
D(massc) = masrte - astrll
D(massa) = airrte/(1.+humid) - wa/wc*astrll
D(ebal) = 0.
if(ihtfl.ne. 0) ! equivalent to adiabatic mixins from TPROP for ihtfl=0
1    D(ebal) = h_masrte*masrte + h_airrte*airrte
1    + h_watrte*watrte - enthalpy*totrteout + surface_a

4 -- sys$desadis:SRC1.FOR

```

```

C
C
      uheff = astrax%L/cc
      sz = 0.
      if(u0 .ne. 0.) sz = ( uheff%alpha1/u0/z0 )*(1./alpha1) * z0
C
C
C
      PRMT(6) = QSTRMX
      prmt(7) = sz
      prmt(8) = hei
      prmt(9) = rho
      prmt(10) = Ri
      prmt(11) = uc
      prmt(12) = ua
      prmt(13) = D(r)
      prmt(16) = wc
      prmt(17) = wa
      prmt(18) = enthalpy
      prmt(19) = temp
      prmt(21) = masrte
      prmt(22) = ht
      prmt(23) = hh
      RETURN
      END
C
C
C.....
C
C      SUBROUTINE FOR OUTPUT FROM SOURCE in the presence of a Blanket
C
C      SUBROUTINE SRC10(TIME,Y,DERY,IHLF,NDIM,PRMT)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C
C      include 'sys$desadis:DEGADIS1.dec'
C
C      COMMON
C      $/ERROR/STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtsc,wteb,wtab,wtuh,XLI,
C      $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
C      $ htcut,ERNOBL,NOBLat,crfser,epsilon
C      $/PARM/ UO,ZO,ZR,HL,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C      $/com_ss/ ess,slen,suid,outcc,outcz,outb,outl,sucl,swal,senl,srhl
C      $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C      $/ALP/ ALPHA,alpha1
C
C      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C      DIMENSION Y(6),DERY(6),PRMT(25)
C
C 5 -- sys$desadis:SRC1.FOR

```

```

C      DIMENSION CURNT(iout_src),BKSP(iout_src),OUTP(iout_src)

C      DATA I/0/,III/0/
C      DATA EMAX/0./,tlast/0./

C      REAL*8 ML,K
C      INTEGER R,mass,massc,massa,ebal
C      DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/

C      data nrec1/0/

C      I = I + 1
C      III = III + 1

C      astr = prnt(6)
C      sz = prnt(7)
C      hei = prnt(8)
C      rho = prnt(9)
C      Ri = prnt(10)
C      uc = prnt(11)
C      ua = prnt(12)
C      vel = prnt(13)
C      prnt(14) = vel
C      if(vel .st. prnt(20)) prnt(20) = -prnt(20)
C      prnt(15) = prnt(16)      ! uc
C      uc = prnt(16)
C      cc = uc * rho
C      ua = prnt(17)
C      enthalpy = prnt(18)
C      temp = prnt(19)
C      prnt(24) = prnt(22)      ! ht
C      prnt(25) = prnt(23)      ! hh

C      IF(hei .Le. 0.0) GO TO 1000

C      QSAV = PI*Y(R)*Y(R)*astr
C      IF(QSAV .LT. EMAX) GO TO 110
C      EMAX = QSAV
C      RM = Y(R)
C      SZM = SZ
110  CONTINUE
C      RMAX = dMAX1(RMAX,Y(R))

C      IF(hei .Le. srcut) GO TO 1000
C      if(cc .le. cclow .and. u0 .eq. 0.) goto 1000      ! no wind
C      if((time.st.tend+1. .and. u0.eq.0. .and. vel.eq.0.)goto 1000!no wind LNG

C      IF(I .NE. 1) GO TO 115
C      CURNT(1) = TIME
C      CURNT(2) = Y(R)
C      CURNT(3) = hei

6 -- sys$de$adis:SRC1.FOR

```

```

CURNT(4) = astr
CURNT(5) = sz
CURNT(6) = yc
CURNT(7) = ya
CURNT(8) = rho
CURNT(9) = ri
CURNT(10) = wc
CURNT(11) = wa
CURNT(12) = enthalpy
CURNT(13) = temp
III = 1
GO TO 125
115 IF(I .EQ. 0) RETURN
C
DO 116 II=1,iout_src
116 BKSP(II) = CURNT(II)
C
CURNT(1) = TIME
CURNT(2) = Y(R)
CURNT(3) = hei
CURNT(4) = astr
CURNT(5) = sz
CURNT(6) = yc
CURNT(7) = ya
CURNT(8) = rho
CURNT(9) = ri
CURNT(10) = wc
CURNT(11) = wa
CURNT(12) = enthalpy
CURNT(13) = temp
C
ERM = 0.
erass = 0.
DO 120 II=2,iout_src
div = curnt(ii)
if(div .eq. 0.) div = srcoer
ER1 = ABS( (CURNT(II)-BKSP(II))/div )
ER2 = ABS( (CURNT(II)-OUTP(II))/div )
if(II.ne.3 .and. ii.ne.9 .and. ii.ne.12 .and. ii.ne.7 .and. ii.ne.11)
1      erass = dMAX1(ER1,ER2,ERMss) ! ex hei,QSTR,Ri,enth,wa,ys for SS
120 ERM = dMAX1(ER1,ER2,ERM)
C
if(check4) then                                ! steady state
    if( .not. (vel.eq. 0..and.time.st.srccs)) goto 124
122      check3 = .true.
           outcc = wc * rho
           swcl = wc
           swal = wa
           srhl = rho
           senl = enthalpy
           outl = sarta * Y(r)

```

7 -- sysdesadis:SRC1.FOR



```

      Qstar = prmt(21)/pi/Y(r)**2
      if(u0.ne. 0.) sz = (alpha1/u0/z0*Qstar*outl/outcc)**(1./alpha1) * z0
      outsz = sz
      outb = outl/2.
      goto 1000

```

```

124      if(ermss .st. srcoer) goto 125

      if( time-tlast .st. srcss) goto 122
      return
      endif

```

```

C      IF(ERM .LT. SRCOER) RETURN
C

```

```

125 CONTINUE
      tlast = time
      DO 130 II=1,iout_src
      IF(III.EQ.1) BKSP(II) = CURNT(II)
130 OUTP(II) = BKSP(II)

```

```

C      III = 0
      NREC1 = NREC1 + 1
      WRITE(9,2000) (OUTP(II),II=1,iout_src)
      RETURN

```

```

C
1000 CONTINUE
      I = -1
      IF(TIME .GE. TEND) CHECK3 = .TRUE.
      NREC1 = NREC1 + 1
      WRITE(lunlos,1100)
      WRITE(lunlos,*) Hei,TIME
      TSC1 = TIME
      if(hei .le. 0.) then
        hei = 0.
        y(r) = dmin1(rmax,y(r))
      endif
      WRITE(9,2000)
      1      TIME,Y(R),hei,ast,sz,yc,ya,rho,ri,wc,wa,enthalpy,teap
      WRITE(lunlos,1110) NREC1

```

```

C      PRMT(5) = 1.
C

```

```

      RETURN
1100  FORMAT(5X,'VALUE OF Hei AT SOURCE TERMINATION -- @ TIME')
1110  FORMAT(5X,'NUMBER OF LINES --> ',I8)
2000  format(1P,16.9,1x,1P,16.9,<iout_src-2>(1x,1P,13.6))
      END

```

```

####

```

```

2 -- sys$desadis:SRC1.FOR

```

```
SUBROUTINE SRTOUT(OPNRUP)
```

```
C
```

```
Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
```

```
include 'sys$desadis:DEGADI93.dec/list'
```

```
C
```

```
COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
$      Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
$      Tsama(maxnob,maxnt),Ttemp(maxnob,maxnt),
$      TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
$      TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
$/SORTIN/TIM(maxnt),NTIM,ISTRT
$/com_sprop/ sas_mw,sas_temp,sas_rhoe,sas_cpk,sas_cpr,
$ sas_ufl,sas_lfl,sas_zsp,sas_name
$/comata/ istab,tamb,pamb, humid, isofl,tsurf, ihtfl,htco,iutfl,wfco
$/com_six/ six_coeff,six_pow,six_min_dist,six_flg
$/alp/ alpha,alpha1
```

```
C
```

```
logical cflag,cflag1
```

```
C
```

```
character*3 sas_name
character*40 OPNRUP
```

```
C
```

```
OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP,CARRIAGECONTROL='FORTRAN')
```

```
C
```

```
WRITE(8,1100)
if(six_flg.eq. 0.) then
    write(8,1102)
else
    write(8,1104)
    write(8,1105) six_coeff,six_pow,six_min_dist
endif
```

```
C
```

```
cflag = isofl.eq. 1.or. ihtfl.eq. 0
cflag1= isofl.eq.1
if(cflag) then
    call adiab(2,wc,wa,sas_lfl,sa,cc_lfl,r,w,t,tt)
    call adiab(2,wc,wa,sas_ufl,sa,cc_ufl,r,w,t,tt)
endif
```

```
C
```

```
DO 110 I=ISTRT,NTIM
```

```
C
```

```
WRITE(8,1119)
WRITE(9,1119)
WRITE(8,1110) TIM(I)
if(cflag1) then
    WRITE(8,1116) (100.*sas_lfl),(100.*sas_ufl),sas_zsp
    WRITE(8,1118)
else
    WRITE(8,1115) (100.*sas_lfl),(100.*sas_ufl),sas_zsp
```

```
1 -- sys$desadis:SRTOUT.FOR
```

```

        WRITE(8,1117)
        endif
WRITE(8,1119)
ip = 0
II = KSUB(I)
C
DO 120 J=1,II
C
    cc = tccstr(J,i)
    rho = Trho(J,i)
    yc = Tyc(J,i)
    temp = Ttemp(J,i)
    gamma = Tgamma(J,i)
    b = tb(J,i)
    sz = tsz(J,i)
    sy = tsy(J,i)
    blfl = 0.
    buf1 = 0.
C
    if(.not.cflas) then
        call adiabat(-2,wc,wa,gas_lfl,wc,cc_lfl,r,w,gamma,tt)
        call adiabat(-2,wc,wa,gas_ufl,wc,cc_ufl,r,w,gamma,tt)
    endif
C
    arg = (gas_zsp/sz)**alpha1
    if(arg .ge. 80.) goto 600
C
    ccz = cc/exp(arg)
    if(ccz .lt. cc_lfl) then
        if(cflas1) then
            WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY
        else
            WRITE(8,1120) TDIST(J,I),yc,Cc,rho,gamma,temp,B,SZ,SY
        endif
        goto 600
    endif
    arg = -(dlog(cc_lfl/cc) + (gas_zsp/sz)**alpha1)
    blfl = sqrt(arg)*sy + b
C
    if(ccz .lt. cc_ufl) then
        if(cflas1) then
            WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY,blfl
        else
            WRITE(8,1120) TDIST(J,I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
        endif
        goto 600
    endif
    arg = -(dlog(cc_ufl/cc) + (gas_zsp/sz)**alpha1)
    buf1 = sqrt(arg)*sy + b
    if(cflas1) then
        WRITE(8,1120) TDIST(J,I),yc,Cc,rho,temp,B,SZ,SY,blfl,buf1
    endif

```

```

      else
        WRITE(8,1120) TDIST(J,I),yc,Cc,rho,sama,temp,B,SZ,SY,blf1,buf1
      endif
C
600 continue
    ip = ip + 1
    if(ip .eq. 3) then
      ip = 0
      write(8,1119)
    endif
120 CONTINUE
110 CONTINUE
C
      CLOSE(UNIT=8)
C
C
1100 FORMAT(1H0,5X,'Sorted values for each specified time.')
1102 format(1H0,5X,'X-Direction correction was NOT applied.')
1104 format(1H0,5X,'X-Direction correction was applied.')
1105 format(1h ,5x,5x,'Coefficient:      ',1p13.5,/,
1       1h ,5x,5x,'Power:              ',1p13.5,/,
1       1h ,5x,5x,'Minimum Distance: ',1p13.5' m')
1110 FORMAT(1H0,5X,'Time after beginning of spill ',G14.7,' sec')
1115 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
1       1 'Concentration',1x,'Density',2x,3x,'Gamma',3x,
1       1 'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
1       1x,'Width to',3x,'Width',/,1x,11x,1x'Fraction',2x,
1       1 11x,11x,11x,11x,3x,'Width',3x,11x,9x,
1       1 2(1p9.3,'mole%',1x),/,1h ,
1       1 99x,4x,'at z= ',0p6.2,' m')
1116 FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
1       1 'Concentration',1x,'Density',2x,
1       1 'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
1       1x,'Width to',3x,'Width',/,1x,11x,1x'Fraction',2x,
1       1 11x,11x,11x,3x,'Width',3x,11x,9x,
1       1 2(1p9.3,'mole%',1x),/,1h ,
1       1 88x,4x,'at z= ',0p6.2,' m')
1117 FORMAT(1H ,4X,'(m)',1x,11x,
1       1 2(1X,'(kg/m**3)',1x),11x,4x,'(K)',
1       1 5(8X,'(m)'))
1118 FORMAT(1H ,4X,'(m)',4x,11x,
1       1 2(1X,'(kg/m**3)',1x),4x,'(K)',
1       1 5(8X,'(m)'))
1119 FORMAT(1H )
1120 FORMAT(1H ,3(1X,1P69.3,1X),2x,0p7.4,2x,1X,1P610.3,1X,
1       1 5(1X,1P69.3,1X))
C
      RETURN
      END
####

3 --.sys$desadis:SRTOUT.FOR

```

```

C.....
C
C
      SUBROUTINE SSG(DIST,Y,D,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      DIMENSION Y(1),D(1),PRMT(1)
C
      include 'sys$desadis:DEGADIS2.dec'
C
      parameter (zero=1.D-10, rcrit=2.5D-3)
C
      COMMON
      $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
      $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wtco
      $/ALP/ ALPHA,alpha1
      $/phicom/ iphifl,delay
C
      REAL*8 K,ML
C
      INTEGER rhoun,dh
      DATA rhoun/1/,dh/2/
C
C*** PRMT(I) I/O
C*** I          VALUE          IN/OUT
C*** --          -----          -
C*** 6          E              IN
C*** 7          Cc             OUT
C*** 8          XU(I)          IN
C*** 9          TO(I)          IN
C*** 10         -              -
C*** 11         NREC(I,2)      OUT -- STARTS OUTPUT UNIT=9
C*** 12         DIST          OUT
C*** 13         sz
C*** 14         yc            out
C*** 15         rho            out
C*** 16         temp           out
C*** 17         samaa          out
C*** 18
C*** 19
C*** 20         rho1ay
C*** 21         sz
C
      XU1 = PRMT(8)
      SY = RT2*DELTA*(DIST + XU1)**BETA
      Erate = PRMT(6)
C
C
      sz0 = prmt(22)

1 -- sys$desadis:SSG.FOR

```

```

      SZ = SZ0
C
C*** MATERIAL BALANCE
C
      III = 0
100  Cc = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/SQRTPI/SY
C
      call adiabat(0,wc,wa,yc,ya,cc,rho,wa,enth,temp)
      cclay = cc/dellay
      call adiabat(0,wc,wa,yclay,ya,cclay,rholay,wal,enth,temlay)
      call addheat(cclay,Y(dh),rholay,temlay,cp)
      Prod = dmax1( Y(rhouh)/rholay/prmt(18), zero)
      SZ = ( Prod ) ** (1./alpha1) * z0
      dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
      if(dif.gt. rcrit) then
         III = III+1
         if(III.gt. 20) call trap(33)
         SZ0 = SZ
         goto 100
      endif
      prmt(20) = rholay
      prmt(21) = SZ
      HEFF = GAMMAF*SZ/ALPHA1
      rit = 0.
C
      temp = temlay
      if(isofl.eq.0 .or. ihtfl.ne.0) then
         rho = dellay*(rholay-rhoa) + rhoa      ! estimate
         temp = (wa/rho)*(rholay*temlay/wal)    ! estimate
         rit = rift(temp,heff)
      endif
C
      RISTR = RIF(RHO,HEFF)
      PHI = PHIF(RISTR,rit)
C
      d(rhouh) = prmt(19)/phi
      heish = heff*dellay
      call surface(temlay,heish,rholay,wal,cp,watrtc,arte)
      if(temp.ge. tsurf .or. temlay.ge. tsub) arte = 0.
      d(dh) = ( arte/dellay-Y(dh)*D(rhouh) )/Y(rhouh)
C
      PRMT(7) = Cc
      PRMT(12) = DIST
      prmt(14) = yc
      prmt(15) = rho
      prmt(16) = temp
      prmt(17) = (rholay-rhoa)/cclay
      RETURN
      END
****

```

```

C.....
C
  SUBROUTINE SSGOUT(X,Y,D,IHLF,NDIM,PRMT)
C
  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

  include 'sys$desadis:DEGADIS2.dec'

C
  parameter (nssg=7, zero=1.e-10)

C
  DIMENSION Y(1),D(1),PRMT(1),BKSP(nssg),OUT(nssg),CURNT(nssg)
C
  COMMON
  $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
  $/comata/ istab,tamb,pamb,humid,isoft,tsurf,ihtfl,htco,iwtfl,wcco
  $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
  $/STOPIT/ TSTOP

C
  REAL*8 K,ML
C
C*** PARAMETER OUTPUT
C
C*** FROM SSG                                OUTPUT TO MODEL
C*** -----                                -----
C*** X                                DIST
C*** PRMT(7)                            Cc
C*** Y(1)                                SZ
C*** prmt(14)                            yc
C*** prmt(15)                            rho
C*** prmt(16)                            temp
C*** prmt(17)                            gamma
C
  ERM = 0.
  T01 = PRMT(9)
  TSL = TS(T01,X)
  prmt(22) = prmt(21)      ! sz
C
  IF (PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR OUTPUT ROUTINE
C
  RII = -100./STPG
  RI = 0.
  CURNT(1) = X
  curnt(2) = prmt(14)      ! yc
  CURNT(3) = PRMT(7)       ! cc
  curnt(4) = prmt(15)      ! rho
  curnt(5) = prmt(17)      ! gamma
  curnt(6) = prmt(16)      ! temp
  CURNT(7) = prmt(21)      ! sz

1 -- sys$desadis:SSGOUT.FOR

```

```

C
  90 CONTINUE
C
C*** RECORD THE CURRENT AND PREVIOUS RECORDS
C
  RI = RI + 1.
C
  DO 100 II=1,nsss
100  bksp(II) = curnt(II)
C
  CURNT(1) = X
  curnt(2) = prmt(14)      ! yc
  CURNT(3) = PRMT(7)       ! cc
  curnt(4) = prmt(15)      ! rho
  curnt(5) = prmt(17)      ! samaa
  curnt(6) = prmt(16)      ! temp
  CURNT(7) = prmt(21)      ! sz
C
C*** stop integration when cc<cclow and time is satisfied
C
  IF(PRMT(7).LT.CcLOW .AND. TSL.GE.TSTOP) GO TO 1000
C
C*** CHECK FOR OUTPUT
C
  DO 110 II=3,nsss
  ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
  ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
110  ERM = dMAX1(ER1,ER2,ERM)
C
C*** OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT
C
  DX = CURNT(1) - OUT(1)
  IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
C
C*** RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE
C*** THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD
C*** THAT POINT AS WELL.
C
  DO 120 II=1,nsss
  IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
120  OUT(II) = BKSP(II)
C
  RI = RII
  PRMT(11) = PRMT(11) + 1.
C
  WRITE(9,*) (OUT(II),II=1,nsss)
  RETURN
C
1000 CONTINUE
C
C*** STOP INTEGRATION

2 -- sys$desadis:SSGOUT.FOR

```



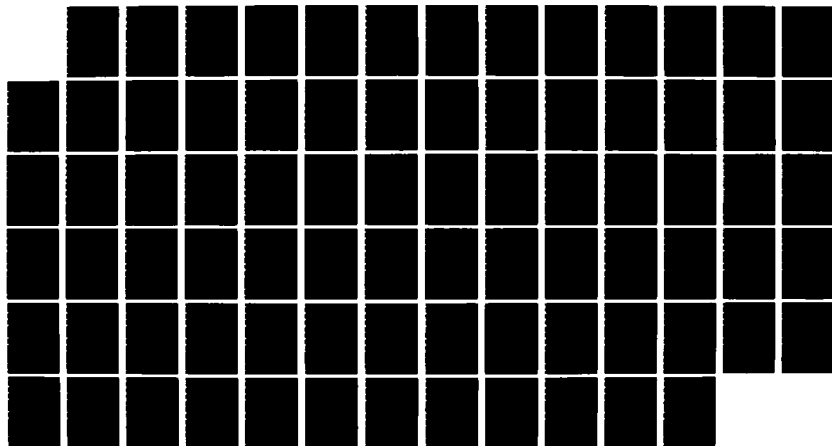
AD-A171 324

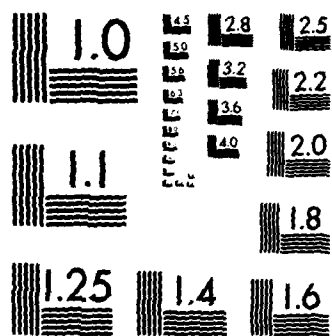
DEVELOPMENT OF AN ATMOSPHERIC DISPERSION MODEL FOR  
HEAVIER-THAN-AIR GAS M. (U) ARKANSAS UNIV FAYETTEVILLE  
DEPT OF CHEMICAL ENGINEERING J A HAVENS ET AL MAY 85  
USCG-D-24-85 DTCG23-80-C-20029 F/G 20/4

3/3

UNCLASSIFIED

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

C-126

```
C
  PRMT(12) = X
  TSTOP = TSL
  PRMT(11) = PRMT(11) + 1.
  WRITE(9,*) (CURNT(II),II=1,nssd)
```

```
C
  PRMT(5) = 1.
```

```
C
  RETURN
  END
```

\*\*\*\*

```

C.....
C
  SUBROUTINE SSGOUT(X,Y,D,IHLF,NDIM,PRMT)
C
  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
  parameter (nssg=9, zero=1.e-10)
C
  DIMENSION Y(1),D(1),PRMT(1),BKSP(nssg),OUT(nssg),CURNT(nssg)
C
  include 'sys$desadis:DEGADIS2.dec/list'
C
  COMMON
  s/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
  s/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
  s/ALP/ALPHA,alpal
C
C
  REAL*8 ML,K
C
  *** PARAMETER OUTPUT
C
  C*** FROM SSG                                OUTPUT TO MODEL
  C*** -----                                -----
  C*** X                                DIST
  C*** PRMT(7)                            Cc
  C*** Y(1)                                SZ
  C*** PRMT(8)                            XU
C
  ERM = 0.
  prmt(22) = prmt(21)
C
  IF(PRMT(11) .NE. 0.) GO TO 90
C
  C*** STARTUP FOR OUTPUT ROUTINE
C
  RII = -100./STPG
  RI = 0.
  CURNT(1) = X
  CURNT(2) = PRMT(14)      ! yc
  CURNT(3) = prmt(7)       ! cc
  CURNT(4) = PRMT(15)      ! rho
  curnt(5) = prmt(17)      ! gamma
  curnt(6) = prmt(16)      ! temp
  curnt(7) = 0.0           ! b
  curnt(8) = prmt(21)      ! sz
  curnt(9) = rt2*delt.*(x+prmt(8))*beta ! sy
C
  90 CONTINUE
C
1 -- sys$desadis:SSGOUTSS.FOR

```

C\*\*\* RECORD THE CURRENT AND PREVIOUS RECORDS

C

RI = RI + 1.

C

```
DO 100 II=1,nss
100 BKSP(II) = CURNT(II)
CURNT(1) = X
CURNT(2) = PRMT(14)      ! yc
CURNT(3) = PRMT(7)       ! cc
CURNT(4) = PRMT(15)      ! rho
CURNT(5) = PRMT(17)      ! sams
CURNT(6) = PRMT(16)      ! temp
CURNT(7) = 0.0           ! b
CURNT(8) = PRMT(21)      ! sz
CURNT(9) = rt2*delta*(x+PRMT(8))*beta ! sy
```

C

C\*\*\* STOP INTEGRATION WHEN Cc < CcLOW

C

IF(PRMT(7).LT.CcLOW) GO TO 1000

C

C\*\*\* CHECK FOR OUTPUT

C

```
DO 110 II=2,nss
if(ii.eq.7) goto 110
ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
110 ERM = dMAX1(ER1,ER2,ERM)
```

C

C\*\*\* OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT

C

```
DX = CURNT(1) - OUT(1)
IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
```

C

C\*\*\* RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE  
C\*\*\* THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD  
C\*\*\* THAT POINT AS WELL.

C

```
DO 120 II=1,nss
IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
120 OUT(II) = BKSP(II)
```

C

```
RI = RII
PRMT(11) = PRMT(11) + 1.
```

C

```
call ssout(out)
RETURN
```

C

1000 CONTINUE

C

C\*\*\* STOP INTEGRATION

C

2 -- sys\$desadis:SSGOUTSS.FOR

C-129

```
PRMT(12) = X  
PRMT(11) = PRMT(11) + 1.
```

C

```
call ssout(curnt)
```

C

```
PRMT(5) = 1.
```

C

```
RETURN  
END
```

####

```

      subroutine ssout(out)
c
      implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension out(9)
c
      common
      /com_sprop/ sas_mw,sas_temp,sas_rhoe,sas_cpk,sas_cpr,
      $ sas_ufl,sas_lfl,sas_zsp,sas_name
      /com_fl/ cflas,cflf,cufl
      /slp/ alphaslp

c
      character*3 sas_name
c
      data ip/0/
c
      logical cflas
c
c
      dist = out(1)
      vc = out(2)
      cc = out(3)
      rho = out(4)
      gamma = out(5)
      temp = out(6)
      b = out(7)
      sz = out(8)
      sy = out(9)
c
      if(.not.cflas) then
         call adiab(-2,vc,wa,sas_lfl,va,cflf,rw,gamma,tt)
         call adiab(-2,vc,wa,sas_ufl,va,cufl,rw,gamma,tt)
      endif
c
      arg = (sas_zsp/sz)**alpha1
      if(arg .ge. 90.) goto 600
c
      ccz = cc/exp(arg)
      if(ccz .lt. cflf) then
         if(cflas) then
            WRITE(8,1120) DIST,vc,Cc,rho,temp,B,SZ,sy
         else
            WRITE(8,1125) DIST,vc,Cc,rho,gamma,temp,B,SZ,sy
         endif
         goto 600
      endif
      arg = -(dlog(cflf/cc) + (sas_zsp/sz)**alpha1)
      blfl = sort(arg)*sy + b
c
      if(ccz .lt. cufl) then
c
1 -- sys$deqadis:SSOUT.FOR

```

C-131

```
      if(cflag) then
WRITE(8,1120) DIST,vc,Cc,rho,temp,B,SZ,SY,bfl
      else
WRITE(8,1125) DIST,vc,Cc,rho,gamma,temp,B,SZ,SY,bfl
      endif
      goto 600
    endif
    arg = -(dlog(cufl/cc) + (gas_zsp/sz)*%alpha1)
    bufl = sqrt(arg)*%v + b
      if(cflag) then
WRITE(8,1120) DIST,vc,Cc,rho,temp,B,SZ,SY,bfl,bufl
      else
WRITE(8,1125) DIST,vc,Cc,rho,gamma,temp,B,SZ,SY,bfl,bufl
      endif
C
600 continue
    ip = ip + 1
    if(ip .eq. 3) then
      ip = 0
      write(8,1119)
    endif
C
C
1119 FORMAT(1H )
1120 FORMAT(1H ,11(1X,1PG9.3,1X))
1125 FORMAT(1H ,3(1X,1PG9.3,1X),2X,0PF7.4,2X,1X,1PG10.3,1X,
1      6(1X,1PG9.3,1X))
C
      return
      end
****
```



```

C .....
C
C   PSEUDO-STEADY STATE SUPERVISOR
C
C   SUBROUTINE SSSUP(H_nsrte)
C
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sysdesadis:DEGADIS2.dec/nolist'
C
C   COMMON
C   $/GEN3/  radg(2,max1),estr(2,max1),srcden(2,max1),srcwc(2,max1),
C   $ srcwc(2,max1),srcenth(2,max1)
C   $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
C   $/GEN1/  ET(2,isen),RiT(2,isen)
C   $/gen2/  den(5,isen)
C   $/com_sprop/  sas_wu,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C   $ sas_wfl,sas_lfl,sas_zfp,sas_name
C   $/PARM/  UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/ERROR/ SYOEP,ERRO,SZOER,MTAIO,MTQOO,MTSZO,ERRP,SNXP,
C   $ WTSZP,WTSP,WTBEP,WDH,ERRG,SNXG,ERTDNF,ERTUPF,UTRUH,WDHG
C   $/comata/ istab,tamb,pamb, humid, isofl,tsurf, ihtfl,htco,iutfl,wtco
C   $/PARMSC/ RM,SM,EMAX,RMAX,TSC1,ALEPH,TEND
C   $/STP/  STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C   $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   $/ALP/  ALPHA,alpha1
C   $/phicom/ i-hifl,dellay
C   $/sprd_con/ ce, delrhomin
C   $/STOPIT/ TSTOP
C   $/CNOBS/ NOBS
C
C   REAL*8 K,ML,L
C   LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   logical Pup,pdn
C
C   character*3 sas_name
C
C   EXTERNAL PSS,PSSOUT,SSG,SSGOUT,OB,OBOUT
C
C   DIMENSION FRMT(22),Y(5),DERY(5),AUX(8,5)
C
C   DATA RTOT/0./
C
C
C   R = AFGEN(RADG,0.0,'RADG')
C   T01 = T0OB(R,0.0)
C   XEND = AFGEN(RADG,TEND,'RADG')
C   TCF = T0OB(-XEND,TEND)
C
C   DT0B = (TOF-T01)/FLOAT(NOBS+1)
C
1 -- sysdesadis:SSSUP.FOR

```

```

      T01 = T01 + DT0B
C
C
      DO 120 I = 1,NOBS
C
C*** RESET AGAIN
C
      AGAIN = .FALSE.
C
      T0(I) = DT0B*float(I-1) + T01
      pup = .true.
      pdn = .true.
C
C*** IF(XEND .GT. XIT(TEND,T0(I))) -- IS
C*** TRUE WHEN THE SOURCE WILL TERMINATE BEFORE THE OBSERVER
C*** CAN REACH THE DOWNWIND EDGE.
C
      IF(XEND .GT. XIT(TEND,T0(I))) then
          pdn = .false.
          TDOWN = TEND
          end if
C
C*** IF(XIT(0.0,T0(I)) .st. -R) -- IS
C*** TRUE WHEN THE SOURCE WILL begin after THE OBSERVER
C*** has passed THE DOWNWIND EDGE @ t=0.0
C
      R = AFGEN(RADG,0.0,'RADG')
      IF(t0(i).le.0. .and. XIT(0.0,T0(I)).st.-R) then
          pup = .false.
          TUP = 0.0
          endif
C
      if(pup) TUP = TUPF(T0(I))
      if(pdn) TDOWN = TDNF(T0(I))
C
      XDOWN = XIT(TDOWN,T0(I))
      XUP = XIT(TUP,T0(I))
      WRITE(1unlos,1160) TUP,XUP,TDOWN,XDOWN
C
C*** SET UP INTEGRATION PARAMETERS FOR EACH OBSERVER.
C
      do ijk=1,22
          prnt(ijk) = 0.
          enddo
      do ijk=1,5
          y(ijk) = 0.
          der_y(ijk) = 0.
          do ikl=1,8
              aux(ijkl,ijk) = 0.
          enddo
      enddo

```

2 -- sys\$desadis:SSSUP.FOR

```

C
  PRMT(1) = TUP
  PRMT(2) = TDOWN
  PRMT(3) = STPO
  PRMT(4) = ERRO
  PRMT(5) = dMAX1(1.D0,(TDOWN-TUP)/50.D0)
  PRMT(6) = TO(I)
  prmt(7) = xup
  PRMT(13) = XDOWN - XUP
C
  Y(1) = 0.
  Y(2) = sz0er      ! Hrate
C
  Y(3) = SZOER
  y(3) = sz0er      ! Crate
  y(4) = 0.
  y(5) = sz0er*H_msrte ! Hrate
C
  DERY(1) = WTAIO
  DERY(2) = WTQOO
  DERY(3) = WTSZO
  DERY(4) = 1.
  DERY(5) = 1.
C
  NDIM = 4
  if(isofl.eq. 0 .and. ihtfl.ne. 0) ndim=5
C
C*** PERFORM INTEGRATION.
C
  WRITE(lunlos,1120) I
1120 FORMAT(/,' Entering Observer Integration Step for Observer # ',
           $I3)
C
  CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,OB,OBOUT,AUX)
C
  IF(IHLF .GE. 10) CALL trap(8,IHLF)
C
  write(lunlos,1125)
1125 format(' ',10x,'Observer Integration complete...')
C
C
  cclay = prmt(14)
  cc = cclay*delay
  uclay = prmt(15)
  uslay = prmt(16)
  enthlay = prmt(17)
  rhelay = prmt(18)

  call setden(uclay,uslay,enthlay)
  if(isofl.eq. 1) goto 200
  do iii= 1:isen
    if(den(1,iii) .gt. 1.) then

```

3 -- sysidesadis:SSSUP.FOR

```

        ii = iii+1
        rho1ay = den(3,iii-1)
        tem1ay = den(5,iii-1)
        if(ii .st. isen .or. iii.le.3) call trap(2)
        cc = cclay*dellay
    if(cc.st. rhoe) then
        write(lunlos,1126) cc,rhoe
1126      format(/,' ',10('***'),//,' cc: ',1p13.5,' is greater',
1          ' than rhoe: ',1p13.5,/, ' ',10('***'),//)
        cc = rhoe
    endif
        rho = cc*(rho1ay-rhoa)/cclay + rhoa
        wc = cc/rho
        u2 = den(2,iii-2)/den(3,iii-2)
        u1 = den(2,iii-1)/den(3,iii-1)
        slope = (wc-u1)/(u2-u1)
        Yc = dmax1(1.00, slope*(den(1,iii-2)-den(1,iii-1))+den(1,iii-1))
        den(1,iii) = Yc
        den(2,iii) = cc
        den(3,iii) = rho
        if(den(3,iii-2).ne. den(3,iii-1)) then
            slope = (1./rho-1./den(3,iii-1))/
1              (1./den(3,iii-2)-1./den(3,iii-1))
            den(4,iii) = slope*(den(4,iii-2)-den(4,iii-1))+den(4,iii-1)
            den(4,iii) = dmax1(h_easrte,den(4,iii))
            den(5,iii) = slope*(den(5,iii-2)-den(5,iii-1))+den(5,iii-1)
            den(5,iii) = dmax1(sas_temp,den(5,iii))
        else
            den(4,iii) = den(4,iii-1)
            den(5,iii) = den(5,iii-1)
        endif
        temp = den(5,iii)
        den(1,ii) = 2. ! end-of-record
c      if(cc.st. rhoe) call trap(31)
        goto 200
    endif
enddo
C
200  L = XDOWN - XUP
     B = Y(1)
     AREA = B*L
     QSTRO = Y(3)/area
     sz0 = (astro*L/cc * alpha1/u0/z0)**(1./alpha1) * z0
C
     ratio1 = u0*sz0/ALPHA1/ Z0**ALPHA1 *Cc /B/astro/L
     ratio = ratio1* sz0**alpha1 * (B + sartpi/2.*sy0er)
     if(ratio.le. 1.) then
         sy0er = (1./(ratio1*sz0**alpha1) - B)*2./sartpi
     else
         sz0 = (1./((B+ sartpi/2.*sy0er)*ratio1))**(1./alpha1)
     endif
4 -- sys$desadis:SSSUP.FOR

```

```

Erate = 2.*astr0*L*b
IF(Cc .GT. RHOE) then
  WRITE(lunlos,1180) QSTR0,SZ0,Cc
  call trap(30)
endif

```

C

C\*\*\* SHOW THE OPERATOR WHAT IS GOING ON

C

```

  WRITE(lunlos,1160) TUP,XUP,TDOWN,XDOWN
  WRITE(lunlos,1170) AREA,L,B
  WRITE(lunlos,1180) QSTR0,SZ0,sy0er
  write(lunlos,1185) wclay,walay,rholay,cclay,tealay
  write(lunlos,1186) wc,rho,cc,temp
1160 FORMAT(/,' TUP: ',1pG13.5,' XUP: ',1pG13.5,' TDOWN: ',
  & 1pG13.5,' XDOWN: ',1pG13.5)
1170 FORMAT(' AREA: ',1pG13.5,' LENGTH: ',1pG13.5,' B: ',1pG13.5)
1180 FORMAT(' TAKEUP FLUX: ',1pG12.5,' SZ0: ',1pG12.5,
  & ' sy0: ',1ps12.5)
1185 format(' wclay: ',1ps12.5,' walay: ',1ps12.5,
  1 ' rholay: ',1ps12.5,' Cclay: ',1ps12.5,/,
  1 ' tealay: ',1ps13.5)
1186 format(' wc: ',1ps12.5,
  1 ' rho: ',1ps12.5,' Cc: ',1ps12.5,' temp: ',1ps12.5)

```

C

C\*\*\* PREPARE FOR PSEUDO-STEADY STATE INTEGRATION.

C

```

  do ijk=1,22
    prnt(ijk) = 0.
  enddo
  do ijk=1,5
    y(ijk) = 0.
    dery(ijk) = 0.
  do ijk1=1,8
    aux(ijk1,ijk) = 0.
  enddo
enddo

```

C

```

  PRMT(1) = XDOWN
  PRMT(2) = 6.023E23
  PRMT(3) = STPP
  PRMT(4) = ERRP
  PRMT(5) = SMXP
  PRMT(6) = Erate
  PRMT(7) = Cc      ! -- OUTPUT
  PRMT(8) = B       ! -- OUTPUT

```

C

C\*\*\* PRMT(9) &amp; PRMT(10) ARE CONSTANTS FOR D(SY) &amp; D(SZ)

C

```

  PRMT(9) = Cc*sqrt(G*ZO/ALPHA1*GAMMAF)*GAMMAF/UO
  PRMT(10) = ZO**ALPHA1*K*USTAR*ALPHA1*ALPHA1/UO
  PRMT(11) = NREC(I,1)

```

S -- sys\$desadis:SSSUP.FOR

```

C   PRMT(12)= DIST AT COMPLETION -- OUTPUT
C   PRMT(13)= T0(I)
C   prmt(14)= yc      ! output
C   prmt(15)= rho      ! output
C   prmt(16)= temp     ! output; not recorded if isofl=1
C   prmt(17)= gamma    ! output; not recorded if isofl=1 .or. ihtfl=0
C   prmt(18)= u0*z0/alpha1
C   prmt(19)= rhoa*k*ustar*alpha1
C   prmt(20)= rho1ay
C   prmt(21)= sz0
C   prmt(22)= sz0

C   Y(1) = rho1ay*prmt(18)*(SZ0/z0)**alpha1      ! rho*ueff*heff
C   Y(2) = SYOER
C   Y(3) = B + sqrt(pi/2.*sy0er
C   y(4) = 0.      ! added heat

C   DERY(1) = WTSZP
C   DERY(2) = WTSYP
C   DERY(3) = WTBEP
C   dery(4) = WTDH

C   NDIM = 4

C   WRITE(lunlos,1130)
1130 FORMAT(' Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
C   CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
C   IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
C   NREC(I,1) = INT(PRMT(11))
C   WRITE(lunlos,1100) NREC(I,1),T0(I)
1100 FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10,' FOR T0='1e-13.5)
C
C   IF(AGAIN) GO TO 119
C
C*** GAUSSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C*** FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
C*** THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C*** CALCULATED AS IF BEFF=sqrt(pi)*SY/2. RETAINING THE LAST VALUE OF Cc IN THE
C*** MATERIAL BALANCE.
C
C   heat = y(4)
C   rho1ay = prmt(20)

6 -- sysfdesadis:SSSUP.FOR

```

```

Cc = PRMT(7)
rhoun = Y(1)
sz = ( rhoun/rholay/prmt(18) )**(1./alpha1) * z0
SYT = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
XT = PRMT(12)
XV(I) = (SYT/RT2/DELTA)**(1./BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
do ijk=1,22
  prmt(ijk) = 0.
enddo
do ijk=1,5
  u(ijk) = 0.
  deru(ijk) = 0.
do ijk1=1,8
  aux(ijk1,ijk) = 0.
enddo
enddo

C
PRMT(1) = XT
PRMT(2) = 6.023E23
PRMT(3) = STPG
PRMT(4) = ERRO
PRMT(5) = SMXG
PRMT(6) = Erate
PRMT(7) = Cc      ! -- OUTPUT
PRMT(8) = XV(I)
PRMT(9) = T0(I)
C
PRMT(10) = 'BLANK'
PRMT(11) = NREC(I,2)
C
PRMT(12) = DIST AT COMPLETION -- OUTPUT
C
prmt(13) = 'blank'
C
prmt(14) = yc      ! output
C
prmt(15) = rho      ! output
C
prmt(16) = temp      ! output
C
prmt(17) = samea      ! output
prmt(18) = u0*z0/alpha1
prmt(19) = rhoa*k*ustar*alpha1
prmt(20) = rhulay
prmt(21) = sz
prmt(22) = sz
C
Y(1) = rhoun
Y(2) = heat
C
DERY(1) = WTRUH
deru(2) = WTDHG
C
NDIM = 2
7 -- sys$desadis:SSSUP.FOR

```

C-139

```
C      WRITE(lunlos,1140)
1140 FORMAT(' Entering Gaussian Stage of Integration ')
C
C*** PERFORM INTEGRATION
C      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
C
C      IF(IHLF .GE. 10) CALL trap(10,IHLF)
C
C      NREC(I,2) = INT(PRMT(11))
C      RTOT = RTOT + FLOAT(NREC(I,1) + NREC(I,2))
C      WRITE(lunlos,1110) RTOT,I
1110 FORMAT(5X,'TOTAL NUMBER OF RECORDS = ',1P613.4,' THROUGH',
  $' OBS # ',I3)
C
C      IF(RTOT .GT. 120000.) CALL trap(11)
C
119 CONTINUE
120 CONTINUE
C
      RETURN
      END
****
```



```

C.....
C
  SUBROUTINE STRT2(OPNRUP,H_ssrte)
C
  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

  include 'sys$desadis:DEGADIS2.dec'
C
  COMMON
    $/GEN3/ rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
    $ srcwa(2,max1),srcenth(2,max1)
    $/TITL/ TITLE
    $/GEN1/ ET(2,isen),R1T(2,isen)
    $/GEN2/ DEN(5,isen)
    $/ITI/ T1,TINP,TSRC,TOBS,TSRT
    $/ERROR/SYOER,ERRO,SZOER,MTAIO,WTQOO,WTSTO,ERRP,SMXP,
    $ WTSZP,WTSP,WTBEP,MTDN,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,MTDHG
    $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
    $/COM_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpp,
    $ sas_uf1,sas_lfl,sas_zsp,sas_name
    $/comata/ istab,tamb,pamb,humid,iso1,tsurf,ihtfl,htco,iutfl,wfco
    $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
    $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
    $/COM_sisx/ sisx_coeff,sisx_pow,sisx_min_dist,sisx_flag
    $/NEND/ POUNDN,POUND
    $/ALP/ ALPHA,alpha1
    $/phicom/ iphifl,delay
    $/sprd_con/ ce,delrhomin
    $/COM_SURF/ HTCUT
C
  character*80 TITLE(4)
C
  character*4 pound
  character*24 TINP,TSRC,TOBS,TSRT
  character*3 sas_name
C
  REAL*8 K,ML
  LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
  character*40 OPNRUP
C
  OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
  DO 90 I = 1,4
    90 READ(9,1000) TITLE(I)
  1000 FORMAT(A80)
C
  READ(9,*) NP
  DO 100 I=1,NP
    100 READ(9,*) ET(1,I),ET(2,I),R1T(2,I)
  1 -- sys$desadis:STRT2.FOR

```

```

      DO 110 I=1,NP
110  R1T(1,I) = ET(1,I)
      I = NP + 1
      ET(1,I) = POUNDN
      ET(2,I) = POUNDN
      R1T(1,I) = POUNDN
      R1T(2,I) = POUNDN

```

C

```

      READ(9,*) NP
      DO 220 I=1,NP
220  READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,i),den(5,i)
      den(1,np+1) = 2.

```

C

```

      READ(9,*) NP
      DO 300 I=1,NP
      READ(9,*) rads(1,I),rads(2,I),astr(2,I),srcden(2,I),srcwc(2,i),
1      srcwa(2,i),srcenth(2,i)
      astr(1,I) = rads(1,I)
      srcden(1,I) = rads(1,I)
      srcwc(1,i) = rads(1,i)
      srcwa(1,i) = rads(1,i)
      srcenth(1,i) = rads(1,i)

```

```

300  continue
      I = NP + 1
      rads(1,I) = POUNDN
      rads(2,I) = POUNDN
      astr(1,I) = POUNDN
      astr(2,I) = POUNDN
      srcden(1,I) = POUNDN
      srcden(2,I) = POUNDN
      srcwc(1,I) = POUNDN
      srcwc(2,I) = POUNDN
      srcwa(1,I) = POUNDN
      srcwa(2,I) = POUNDN
      srcenth(1,I) = POUNDN
      srcenth(2,I) = POUNDN

```

C

```

      READ(9,1010) TINP,TSRC
      READ(9,1010) tobs,TSRT

```

C

```

      READ(9,*) U0,Z0,ZR,ML,USTAR
      read(9,*) K,G,RHOE,RHOA,DELTA
      read(9,*) BETA,GAMMAF,CcLOW

```

C

```

      READ(9,*) RH,SZM,EMAX,RMAX,TSC1
      read(9,*) ALEPH,TEND

```

C

```

      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5

```

C

```

      READ(9,*) ALPHA
      alpha1 = alpha + 1.

```

```

2 -- sysidesadis:STRT2.FOR

```

```

C
  read(9,1020) sas_name
  read(9,*) sas_aw,sas_temp,sas_rhoe
  read(9,*) sas_cpk,sas_cpf
  read(9,*) sas_ufl,sas_lfl,sas_zsp
C
  read(9,*) istab
  read(9,*) tsab,psab, humid
  read(9,*) isofl,tsurf
  read(9,*) ihtfl,htco
  read(9,*) iutfl,utco
C
  read(9,*) six_coeff,six_pow,six_min_dist
C
  read(9,*) ihifl,dellay
C
  H_masrte = 0.
  if(isofl.eq. 0) read(9,*) H_masrte
C
  READ(9,*) HTCUT, ce, delrhomin
C
1010 format(2(a24,1x))
1020 format(a3)
C
  CLOSE(UNIT=9)
C
  RETURN
  END
####

```

```

C.....
C
      SUBROUTINE STRT2(OPNRUP,H_msrte)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      INCLUDE 'sys$desadis:DEGADIS2.DEC/LIST'
C.....
C
C      BLOCK COMMON
C
      COMMON
      $/TITL/ TITLE
      $/GEN2/ DEN(5,IGEN)
      $/ITI/ T1,TINP,TSRC,TOBS
      $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
      $/com_ss/ ESS,SLEN,SWID,OUTCC,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
      $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      $/com_sprop/ sas_sw,sas_temp,sas_rhoe,sas_cpk,sas_cpp,
      $ sas_ufl,sas_lfl,sas_zsp,sas_name
      $/comata/ istab,tamb,ramb,humid,isofl,tsurf,ihtfl,htco,iutfl,wtco
      $/NEND/ POUNDN,POUND
      $/ALP/ ALPHA,alpha1
      $/phicom/ i-phifl,dellay
      $/sprd_con/ ce, delrhomin
      $/COM_SURF/ HTCUT
C
      character*80 TITLE(4)
      character*24 TSRC,TINP,TOBS
      character*40 OPNRUP
      character*3 sas_name
      character*4 pound
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      DO 90 I = 1,4
      90 READ(9,1000) TITLE(I)
      1000 FORMAT(A80)
C
      read(9,*) np
      do 100 i=1,np
      100 read(9,*) dummy1,dummy2,dummy3
C
      READ(9,*) NP
      DO 120 I=1,NP
      120 READ(9,*) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
      I = NP + 1
1 -- sys$desadis:STRT2SS.FOR

```

```

DEN(1,I) = 2.
C
  read(9,*) np
  do 140 i=1,np
140 read(9,*) dummy1,dummy2,dummy3,dummy4,dum5,dum6,dum7
C
  read(9,1100) tinp,tsc
  read(9,1100) tobs,tsrt
1100 format(a24,1x,a24)
C
  READ(9,*) U0,Z0,ZR,HL,USTAR
  Read(9,*) K,G,RHOE,RHOA,DELTA
  read(9,*) BETA,GAMMAF,CcLOW
C
  read(9,*) dummy1
  read(9,*) dummy1
C
  READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
  READ(9,*) ALPHA
  alpha1 = alpha + 1.
C
  read(9,1200) sas_name
  read(9,*) sas_aw,sas_temp,sas_rhoe
  read(9,*) sas_cpk,sas_crp
  read(9,*) sas_ufl,sas_lfl,sas_zsr
C
  read(9,*) istab
  read(9,*) tamb,psmb,humid
  read(9,*) isofl,tsurf
  read(9,*) ihtfl,htco
  read(9,*) iutfl,utco
C
  read(9,*) dummy1
C
  READ(9,*) ESS,SLEN,SWID
  read(9,*) OUTCc,OUTSZ,OUTB,OUTL
  read(9,*) sucl,sual,senl,srhl
C
  read(9,*) iphifl,dellay
C
  h_masrte = 0.
  if(isofl.eq. 0) read(9,*) H_masrte
C
  READ(9,*) HTCUT, ce, delrhomin
C
  CLOSE(UNIT=9)
  RETURN
1200 format(a3)
  END
####
2 -- sys$desadis:STRT2SS.FOR

```

```

C.....
C
  SUBROUTINE STRT3(OPNRUP)
C
  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

  include 'sys$desad.s:DEGADIS3.dec/list'

C
C  BLOCK COMMON
C
  COMMON
    $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
    $/GEN2/ DEN(5,isen)
    $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
    $/COM_PROP/ sas_sw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
    $ sas_ufl,sas_lfl,sas_zsp,sas_name
    $/ITI/ TI,TINP,TSRC,TOBS,TSRT
    $/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iutfl,wtco
    $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
    $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
    $/com_six/ six_coeff,six_pow,six_min_dist,six_flg
    $/NEND/ POUNDN,POUND
    $/ALP/ ALPIA,alpha1
    $/CNOBS/ NOBS
C
    character*3 sas_name
    character*40 OPNRUP
    character*24 TINP,TSRC,TOBS,TSRT
C
    REAL*8 K,ML
    LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
    OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
    READ(9,*) NOBS
    DO 125 I=1,NOBS
125 READ(9,*) NREC(I,1),NREC(I,2),T0(I),XV(I)
C
    READ(9,*) Npts
    DO 140 I=1,Npts
140 READ(9,*) den(1,i),den(2,i),den(3,i),den(4,i),den(5,i)
    den(1,npts+1) = 2.
C
    READ(9,*) UO,ZO,ZR,ML,USTAR
    read(9,*) K,G,RHOE,RHOA,DELTA
    read(9,*) BETA,GAMMAF,CcLOW
C
    READ(9,1010) TINP,TSRC
    read(9,1010) TOBS,TSRT
1010 format(2(a24,1x))

1 -- sys$desadis:STRT3.FOR

```

```

C      READ(9,*) RH,SZM,EMAX,RMAX,TSC1
      read(9,*) ALEPH,TEND
C
      read(9,1020) sas_name
      read(9,*) sas_aw,sas_temp,sas_rhoe
      read(9,*) sas_cpk,sas_cpr
      read(9,*) sas_ufl,sas_lfl,sas_zsp
      read(9,*) istab
      read(9,*) tamb,pamb,humid
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
      read(9,*) six_ccoeff,six_pow,six_min_dist
1020 format(a3)
C
      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      READ(9,*) ALPHA
      alpha1 = alpha + 1.
C
      CLOSE(UNIT=9)
C
      RETURN
      END
****

```

```

C      Surface effects
C
C      SUBROUTINE Surface(temp,height,rho,mole,cp,watrtc,arte)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'sys$desadis:DEGADIS1.dec'
C
C      COMMON
C      $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C      $/COMATA/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtf1,utco
C      $/ALP/ ALPHA,alpha1
C      $/PHICOM/ iphif1,delay
C      $/COM_SURF/ HTCUT
C
C      REAL*8 ML,K
C      REAL*8 L,watrtc,mole
C
C      vapor_p(txxx) = 6.0298e-3* exp(5407. *(1./273.15- 1./txxx))
C
C
C      watrtc = 0.
C      arte = 0.
C      if(isofl.eq.1 .or. ihtfl.eq.0) return
C      if(height.le. htcut) return
C      delta_t = tsurf - temp
C      if(delta_t .lt. 0.) return
C      top_vel = u0 * (height/z0)**alpha
C      u10 = u0*(10./z0)**alpha
C      prod_nat = ((rho/mole)**2 * abs(delta_t)) ** 0.333333
C      if(ihtfl .eq. 1) then                ! local correlation
C          hn = 18. * prod_nat
C          hf = 0.
C          if(top_vel .ne.0.) hf = 1.22 * rho*cp * ustar**2/top_vel
C          hf = 1.22 * rho*cp * (ustar/u10)**2*top_vel
C          ho = dmax1(hn,hf)
C      else if(ihtfl .eq. 2) then            ! LLNL correlation
C          ho = htco* rho* cp
C      else if(ihtfl .eq. 3) then            ! Colenbrander's method
C          av_temp = (tsurf + temp)/2.
C          hn = 89.*(delta_t/av_temp**2)**.33333
C          hf = 1.22 * rho*cp * ustar**2/u10
C          ho = dmax1(hn,hf)
C      else
C          ho = htco                ! ihtfl=-1
C      endif
C      arte = ho * delta_t
C
C      i -- sys$desadis:SURFACE.FOR

```



```

      if(arte .lt. 0.) then
        write(6,8000) arte,ho,delta_t
8000      format(' SURFACE? ht < 0.0; ',1p13.5,2x,'ho: ',
1         1p13.5,/, ' delta_t: ',1p13.5)
        call trap(4)
        endif
      watrte = 0.
      if(iwtfl .eq. 0) return
      fo = wtco
        if(iwtfl .gt. 0) then
          fn = 9.9e-3 * prod_nat
          ff = 20.7 * ho /cp /mole
          fo = dmax1(fn,ff)
        endif
      watrte = fo/pamb * (vapor_p(tsurf) - vapor_p(temp))
c
      return
end
****

```

```

C.....
C
C    FUNCTION TO RETURN SZO CALCULATED over the source without
C    a blanket present underneath
C
C    NOTE: Uses the integration package RKGST and cannot
C          be used with any other routine without a local copy
C          of RKGST.
C
C    subroutine SZF(Q,L,sz,cclay,wclay,rholay)
C
C
C    Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C    external szlocal,szloco
C
C    REAL*8 L
C
C    include 'sys$desadis:DEGADIS1.dec'
C
C    COMMON
C    $/szfc/ szstp0,szerr,szstpmx,szsz0
C
C    dimension Y(2),D(2),PRMT(17),aux(8,2)
C
C
C    prmt(1) = 0.
C    prmt(2) = L
C    prmt(3) = szstp0
C    prmt(4) = szerr
C    prmt(5) = szstpmx
C    prmt(6) = Q
C
C    Y(1) = szsz0      ! rho*del*uo*z0/(1.+alpha)*(sz/z0)**(1.+alpha)
C
C    ndim = 1
C
C    call rkst(prmt,y,d,ndim,ihlf,szlocal,szloco,aux)
C
C    if(ihlf.ge. 10) call trap(3,ihlf)
C
C    cclay = prmt(13)
C    wclay = prmt(14)
C    rholay = prmt(15)
C    cc = prmt(16)
C    sz = prmt(17)
C
C    RETURN
C    END
C
1 -- sys$desadis:SZF.FOR

```

```

C
C
C      subroutine szlocal(x,y,d,prnt)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      dimension y(1),d(1),prnt(1)
C
C      common
C      $/parm/ u0,z0,zr,al,ustar,k,g,rho0,rho0,delta,beta,sammaf,cclow
C      $/alp/ alpha,alpha1
C      $/phicom/ iphif1,dellay
C
C      real*8 al,k
C
C      integer rhouhlay/1/
C
C      Q = prnt(6)
C      uclay = Q*x/Y(rhouhlay)
C
C      call adiab(1,uclay,walay,yc,ya,cclay,rholay,wa,enth,temp)
C      cc = cclay*dellay
C      call adiab(0,wc,wa,yc,ya,cc,rho,wa,enth,temp) ! centerline
C
C      uheff = Y(rhouhlay)/rholay/dellay
C
C      sz = ( uheff/u0/z0*(alpha1) )**(1./alpha1) * z0
C      heff = sammaf*sz/alpha1
C      ristar = rif(rho,heff)
C      phi = phif(ristar,0.)
C      wel = dellay * k*ustar*alpha1/phi
C      D(rhouhlay) = wel*rho0 + Q
C
C      prnt(8) = cclay
C      prnt(9) = uclay
C      prnt(10) = rholay
C      prnt(11) = cc
C      prnt(12) = sz
C
C      return
C      end
C
C
C
C      subroutine szloco(x,y, dery,ihlf,ndia, prnt)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      dimension x(1),y(1), dery(1), prnt(1)
C
C 2 -- sys$desadis:SZF.FOR

```

C-151

C

```
prnt(13) = prnt(8)
prnt(14) = prnt(9)
prnt(15) = prnt(10)
prnt(16) = prnt(11)
prnt(17) = prnt(12)
return
end
```

\*\*\*\*

```

subroutine tprop(ifl,wc,wa,enthalpy,vc,va,wa,temp,rho,cp)
C
C  subroutine to return:
C      mole fractions (y's)
C      molecular weight (wm)
C      temperature (temp[=]K)
C      density (rho[=]kg/m**3)
C      heat capacity (cp[=]J/kg/K)
C
C  for a mixture from:
C      mass fractions (w's)
C      enthalpy (J/kg)          for ifl.ne.0
C
C      adiabatic mixing of:   emitted gas @ gas_temp
C                           entrained ambient humid air @ tamb
C                           entrained water from surface @ tsurf
C                           for ifl.eq.0 calculate and return
C
C      adiabatic lookup CALL ADIABAT
C                           for isofl.eq.1 .or. ihtfl.eq.0.and.ifl.ne.1
C
C
C  Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C  include 'sys$dasadis:DEGADIS1.dec'
C
C  parameter (tfrac=0.618, tfraci=1.-tfrac,
C            1      rcrit=0.0005, acrit=1., zero=1.e-20)
C
C  common
C  $/GEN2/ DEN(5,isen)
C  $/com_sprop/ gas_wm,gas_temp,gas_rho,gas_cp,gas_cpv,
C  $ gas_ufl,gas_lfl,gas_xsp,gas_name
C  $/cometa/ istab,tamb,rsab,humid,isofl,tsurf,ihtfl,htco,iutfl,wfco
C
C  character*3 gas_name
C
C*** data for air/water sys
C
C      data wa/28.96/      ! molecular weight of air
C      data ww/18./        ! molecular weight of water
C      data rho_water/1000./ ! liquid water density [=] kg/m**3
C      data cp/1.0063e3/    ! heat capacity of air [=] J/kg/K
C      data cpw/1865./      ! heat capacity of water vapor[=]J/kg/K
C      data dhvap/2.5023e6/  ! latent heat of vap [=]J/kg water
C      data dhfus/0.33e6/    ! latent heat of fus [=]J/kg water
C
C  logical rev
C
C
C      vapor_p(txxx) = 5.0298e-3* exp(5407. * (1./273.15- 1./txxx))
C
1 -- sys$dasadis:TPROP.FOR

```

```

      sat_hum(p_tot,p_vp) = 0.622* p_vp/ (p_tot- p_vp) ! ks w/ks BDA
      rho_wair(p_tot,humxx) =
      1      (.00293+ .00456*humxx)/p_tot/(1.+humxx) ! m**3/ks /K
c
c
      ww = 1.-wc-wa
      wa = 1./(wc/gas_wu + wa/waa + ww/waw)
      yc = wa/gas_wu *wc
      ya = wa/waa *wa
c
c
      if(isofl.eq. 1) then
        call adiab(1,wc,wa,yc,ya,cc,rho,wa,enthalpy,temp)
        return      ! interp density from wc
      endif
c
c
      if(ifl.eq. 0)
      1  enthalpy = wc*cc*(gas_temp)*(gas_temp - tamb)
      1      + (ww - wa*humid)*cpw*(tsurf - tamb)
c      1      + wa*(1.+humid)*cpa*(tamb - tamb)      ! TR=tamb
c
c
c
      if(ifl.eq. 1 .and. ihtfl.eq.0) then
        call adiab(1,wc,wa,yc,ya,cc,rho,wa,enthalpy,temp)
        return      ! interp density from wc
      endif
c
c
      rev = .false.
100  continue
      tmin = dmin1(gas_temp,tsurf)
      tmin0 = tmin
      tmax = dmax1(tsurf,tamb)
      tmax0 = tmax
      temp = (tmin+tmax)/2.
c
      do 300 j=1,35
        guess = enthal(wc,wa,temp)
        dif = enthalpy - guess
        sum = (abs(enthalpy) + abs(guess))/2. + zero
        if(abs(dif)/sum.le.rcrit .or. abs(dif).le.acrit) goto 400
c
        if(dif.lt. 0.) then
          if(rev) tmax = temp
          if(.not.rev) tmin = temp
          temp = tmin + (tmax-tmin) * tfrac
        else
          if(rev) tmin = temp
          if(.not.rev) tmax = temp
        end if
      end do
400  -- sys$de$adis:TPROP.FOR

```

```

      temp = tmin + (tmax-tmin) * tfrac1
    endif

c
300 continue
    rev = .not.rev
    if(rev) goto 100
    write(lunlog,8050) wc,wa,enthalpy
8050 format(' TPROP? wc: ',1p12.5,1x,
1      ' wa: ',1p12.5,1x,'enthalpy: ',1p12.5)
    elow = enthal(wc,wa,tain0)
    if(enthalpy.lt. elow) then      ! catch out of bounds numbers
        temp = tain0
        enthalpy = elow
        goto 400
    endif
    elow = enthal(wc,wa,tmax0)
    if(enthalpy.gt. elow) then
        temp = tmax0
        enthalpy = elow
        goto 400
    endif

c
    call trap(24)

c
c
400 continue      ! density calculation
    vp = vapor_p(temp)
    sat = dain1( sat_hum(pamb,vp), humid)
    rho = 1./((temp*ropw_air(pamb,sat)*wa*(1.+sat)
1      + wc*temp/sas_temp/sas_rhoe + (ww-wa*sat)/rho_water)

c
    tmin = temp + 10.
    if(tmin .gt. tmax0) tmin = temp - 10.
    if(tmin .lt. tmin0) tmin = temp + .1

c
    tmax = enthal(wc,wa,tmin)
    cp = (enthalpy - tmax)/(temp - tmin)
    if(cp .lt. cps) cp = cps      ! nominal value of air

c
    return
end

c
c
function cpc(temp)

c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      common
      %/com_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpe,
      % sas_ufl,sas_lfl,sas_zsp,sas_nene

3 -- sys$de$adis:TPROP.FOR

```

```

C      data con/3.33e4/
C
C      character*3 sas_name
C
C      cpc = con
C      if(temp .ne. sas_temp) then
C          cpc = con + sas_cpk*
C      1 (temp**sas_cpf - sas_temp**sas_cpf)/(temp- sas_temp)
C      endif
C      cpc = cpc/sas_aw
C      return
C      end
C
C
C      function enthal(wc,wa,temp)      ! used by TPROP
C
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      parameter (delta=10.)
C
C      common
C      /com_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C      $ sas_ufl,sas_lfl,sas_zsp,sas_name
C      /comata/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtf,wtco
C
C      character*3 sas_name
C
C      data cps/1006.3/      ! heat capacity of air [=]J/kg/K
C      data cpw/1865./      ! heat capacity of water vapor[=]J/kg/K
C      data dhvap/2.5023e6/  ! latent heat of vap [=]J/kg water
C      data dhfus/0.33e6/   ! latent heat of fus [=]J/kg water
C
C
C      ww = 1.-wa-wc
C      vp = 6.0298e-3* exp(5407. *(1./273.15- 1./temp))
C      sat = 0.622 * vp/ (pamb - vp) ! kg w/kg BDA
C      wustar = wa * sat
C
C      dh = dhvap
C      frac = 0.
C      if(temp .lt. 273.15) frac = dmini( (273.15D0-temp)/delta,1.D0)
C      dh = dhvap + dhfus*frac
C      if(wa .eq. 0.) goto 1000
C      cloud_hum = ww/wa
C      if(cloud_hum.le. sat) dh = 0.0
C
C
C      1000 enthal = wc*cpc(temp)*(temp - tamb)
C      1      - dmax1((ww- wustar),0.D0)*dh
C      1      + (ww- wa*humid)*cpw*(temp - tamb)
C
C
C      4 -- sys$desadis:TPROP.FOR

```



```

1      + wa*(1+humid)*cpa*(temp - tamb)      ! TR=tamb
C
return
end
C
C
subroutine adiabatic( ifl,wc,wa,yc,ys,cc,rho,wm,enthalpy,temp)
C
C  subroutine to return:
C      mass fractions (w's)
C      mole fractions (y's)
C      concentration (cc [=] kg/m**3)
C      density (rho [=] kg/m**3)
C      molecular weight (wm)
C      enthalpy ( [=] J/kg)
C      temperature (temp [=] K)
C
C  for a mixture from DEN lookup of adiabatic mixing calculation
C  den(1,i)      mole fraction (yc)
C  den(2,i)      concentration (cc [=] kg c/m**3)
C  den(3,i)      mixture density (rho [=] kg mix/m**3)
C  den(4,i)      mixture enthalpy (enthalpy [=] J/kg)
C  den(5,i)      mixture temperature (temp [=] K)
C
C  ifl indicates given information:
C  -2) mole fraction (Yc) and assumption of constant gamma in enthalpy
C  -1) concentration (cc) and assumption of constant gamma in enthalpy
C  0) concentration (cc)
C  1) mass fraction c (wc)
C  2) mole fraction (Yc)
C
C
Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

include 'sys$desadis:DEGADISIN.dec'
C
common
$/GEN2/ DEN(5,isen)
$/para/ u0,z0,zr,ml,ustar,k,s,rho,rho,delta,beta,gammasf,cclo
$/com_sprop/ sas_aw,sas_temp,sas_rho,sas_cpk,sas_cpf,
$ sas_ufl,sas_lfl,sas_zsp,sas_name
$/comata/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iutfl,utco
C
character*3 sas_name
real*8 ml,k
C
c*** data for air/water sys
C
data wa/29.96/      ! molecular weight of air
data ww/18./        ! molecular weight of water
C
5 -- sys$desadis:TPROP.FOR

```

```

c
  if(ifl.ne. 0) goto 1000
  ccl = cc
  if(cc.lt. 0.) ccl=0.
    i = 2
30    if(den(1,i).st. 1.) then
      i=i-1
      if(cc.st. den(2,i)) ccl=den(2,i)
      goto 50
    endif
    if(cc.le. den(2,i)) goto 50      ! lookup in concentration
    i=i+1
    goto 30
50  slope = (den(3,i)-den(3,i-1)) / (den(2,i)-den(2,i-1)) ! interr in conc
    rho = (ccl - den(2,i-1))*slope + den(3,i-1)
    wcl = ccl / rho
    wc = wcl
    wa = (1.-wc)/(1.+humid)      ! no choice with given information
    ww = humid*wa
    wm = 1./(wc/gas_mw + wa/waa + ww/wmw)
    yc = wm/gas_mw * wc
    ya = wm/waa * wa
    goto 8000

```

```

c
c
1000 if(ifl.ne. -1) goto 1500
    ccl = cc
    if(ccl.lt. 0.) ccl=0.
    samms = enthalpy
    wc = ccl/(rho+ccl*samms)
    wa = (1.-wc)/(1.+humid)      ! no choice with given information
    ww = 1.-wa-wc
    wm = 1./(wc/gas_mw + wa/waa + ww/wmw)
    yc = wm/gas_mw * wc
    ya = wm/waa * wa
    rho = ccl/wc
    return

```

```

c
c
1500 if(ifl.ne. -2) goto 1700
    ycl = yc
    if(ycl.lt. 0.) ycl=0.
    samms = enthalpy
    yw = (1.-ya-ycl)
    ym = yc*gas_mw + ya*wa + yw*wmw
    wc = gas_mw/ym * ycl
    wa = waa/ym * ya
    cc = wc*rhoa/(1. - samms*wc)
    rho = cc/wc
    return

```

```

c
5 -- systdesadis:TPROP.FOR

```

```

c
1700 if(ifl.ne. 2) goto 2000
    ycl = yc
    if(yc .lt. 0.) then
        ycl = 0.
        wa = 1./(1.+humid)
        ww = 1.-wa
        wm = 1./(was/wa + waw/ww)
        y3 = wa/was * wa
    endif
    if(yc .st. 1.) then
        ycl = 1.
        ya = 0.
    endif
    i = 2
1730 if(den(1,i) .st. 1.) then
        i = i-1
        goto 1750          ! extrapolate
    endif
    if(yc.le. den(1,i)) goto 1750  ! lookup in mole frac
    i=i+1
    goto 1730
1750 slope = (den(2,i)-den(2,i-1)) / (den(1,i)-den(1,i-1))  ! interr in y
        cc = (ycl - den(1,i-1)) * slope + den(2,i-1)
        slope = (den(3,i)-den(3,i-1)) / (den(1,i)-den(1,i-1)) ! interr in y
        rho = (ycl - den(1,i-1)) * slope + den(3,i-1)

        wc = cc/rho
        wm = ycl*was_wa + ya*wa + (1.-ycl-ya)*waw
        wa = ya*was/wa

        i = 2
1760 if(den(1,i) .st. 1.) then
        i = i-1
        goto 1800          ! extrapolate
    endif
    cwc = den(2,i)/den(3,i)
    if(wc.le. cwc) goto 1800      ! lookup in mass frac
    i=i+1
    goto 1760
1800    w1 = den(2,i-1)/den(3,i-1)
        w2 = den(2,i)/den(3,i)
        slope = (den(4,i)-den(4,i-1)) / (w2 - w1)          ! interr in w
        enth3py = (wc - w1) * slope + den(4,i-1)
        slope = (den(5,i)-den(5,i-1)) / (w2 - w1)          ! interr in w
        temp = (wc - w1) * slope + den(5,i-1)
c
        return
c
c
2000 if(ifl.ne. 1) goto 9000

7 -- sys$desadis:TPROP.FOR

```

```

wcl = wc
  if(wc .lt. 0.) then
    wcl = 0.
    wa = 1./(1.+humid)
  endif
  if(wc .st. 1.) then
    wcl = 1.
    wa = 0.
  endif
wu = 1.-wa-wcl
wa = 1./(wcl/sas_wu + wa/was + wu/waw)
wc = wa/sas_wu *wcl
ws = wa/was *wa
i = 2
2030 if(den(1,i) .st. 1.) then
  i = i-1
  goto 2050          ! extrapolate
endif
if(wc.le. den(1,i)) goto 2050  ! lookup in mole frac
i=i+1
goto 2030
2050 slope = (den(3,i)-den(3,i-1)) / (den(1,i)-den(1,i-1))
rho = (wc-den(1,i-1))*slope + den(3,i-1)
slope = (den(2,i)-den(2,i-1)) / (den(1,i)-den(1,i-1))
cc = (wc-den(1,i-1))*slope + den(2,i-1)
i = 2
2060 if(den(1,i) .st. 1.) then
  i = i-1
  goto 8000          ! extrapolate
endif
cwc = den(2,i)/den(3,i)
if(wcl.le. cwc) goto 8000    ! lookup in mass frac
i=i+1
goto 2060

c
c
8000      w1 = den(2,i-1)/den(3,i-1)
          w2 = den(2,i)/den(3,i)
          slope = (den(4,i)-den(4,i-1)) / (w2 - w1)          ! interp in w
          enthalpy = (wcl - w1) *slope + den(4,i-1)
          slope = (den(5,i)-den(5,i-1)) / (w2 - w1)          ! interp in w
          temp = (wcl - w1) *slope + den(5,i-1)

c
      return
c
9000 call trap(26)
      end

c
c
      subroutine setenthal(h_masrte,h_airrte,h_watrte)
c
c -- sys$desadis:TPROP.FOR

```

```

c      subroutine to load /com_ENTHAL/ through passed arguments if needed
c
c
c      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
c
c      include 'sys$desadis:DEBADISIN.dec'
c
c      common
c      $/com_sprop/ sas_aw, sas_temp, sas_rhoe, sas_cpk, sas_cpf,
c      $ sas_ufl, sas_lfl, sas_zsp, sas_name
c      $/comata/ istab, tamb, pamb, humid, isofl, tsurf, ihtfl, htco, iwtfl, wtco
c
c      character*3 sas_name
c
c*** data for air/water sys
c
c      data cpa/1.0063e3/      ! heat capacity of air [=]J/kg/K
c      data cpw/1865./        ! heat capacity of water vapor [=]J/kg/K
c
c      h_aasrte = 0.
c      h_airrte = 0.
c      h_watrte = 0.
c
c      if(isofl.eq. 1) return
c
c      h_aasrte = cpc(sas_temp)*(sas_temp - tamb)      ! TR=tamb
c
c      h_airrte = (1.+humid)*cpa*(tamb - tamb) = 0.
c
c      if(iwtfl .eq. 0) return
c      h_watrte = cpw*(tsurf - tamb)
c
c      return
c      end
c
c
c
c      subroutine setden(wc, wa, enthalpy)
c
c      subroutine to load /GEN2/ as needed
c
c      adiabatic mixing of:      WC
c                               WA
c                               WW @ specified enthalpy
c
c                               with ambient humid air @ tamb
c
c      den(1,i)      mole fraction (yc)
c      den(2,i)      concentration (cc [=] kg c/m**3)
c      den(3,i)      mixture density (rho [=] kg mix/m**3)
c
c 9 -- sys$desadis:TPROP.FOR

```

## C-161

```

c      den(4,i)      mixture enthalpy (enthalpy [=] J/kg)
c      den(5,i)      mixture temperature (temp [=] K)
c
c
c      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c      include 'sys$desadis:DEGADISIN.dec'

c      parameter (tcrit=0.002, zero=1.e-20)
c      parameter (iils=200, ils=iils-1, iback=25)

c      common
c      $/GEN2/ DEN(5,isen)
c      $/COM_PROP/ $as_aw,$as_temp,$as_rhoe,$as_cpk,$as_cpf,
c      $ $as_ufl,$as_lfl,$as_zsp,$as_name
c      $/COMETA/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwfl,wtco

c      character*3 $as_name

c      dimension curnt(5),backsp(5,iback)

c
c*** data for air/water sys
c
c      data wwa/28.96/      ! molecular weight of air
c      data wwv/18./        ! molecular weight of water
c      data cpa/1.0063e3/    ! heat capacity of air [=]J/kg/K
c      data cpw/1865./      ! heat capacity of water vapor[=]J/kg/K
c
c
c
c      if(isofl.eq. 1) return

c
c      k = 1
c      den(1,k) = 0.0        ! yc
c      den(2,k) = 0.0        ! cc
c      den(3,k) = pamb*(1.+humid)/(.00283+ .00456*humid)/tamb ! rhoa
c      den(4,k) = 0.0        ! enthalpy of ambient air! TR=tamb
c      den(5,k) = tamb

c
c
c      do 300 i= ils,1,-1
c      zbda = (float(i)/float(iils)) / (1.+humid)
c      zw = zbda*humid
c      zs = 1.-zbda-zw

c
c      enmix = zs*enthalpy + zbda*(1.+humid)*cpa*tamb ! TR=tamb
c      enmix = zs*enthalpy

c
c      zbda = zbda + zs*wa

10 -- sys$desadis:TPROP.FOR

```

```

      zs =      zs$uc
      call tprop(2,zs,zbda,ennix,vcysa,wa,teap,rho,ca)
      cc = zs$rho
c
c
      curnt(1) = yc
      curnt(2) = cc
      curnt(3) = rho
      curnt(4) = ennix
      curnt(5) = teap
c
      if(i .eq. ils) then
         ind = 1
         do 150 JJ= 1,5
150           backsp(JJ,ind) = curnt(JJ)
              soto 300
         endif
c
c      ADIABAT interpolation scheme
c
      err = 0.
      do 180 iind = 1,ind
         yc = backsp(1,iind)
         cc = backsp(2,iind)
         rho = backsp(3,iind)
         ennix = backsp(4,iind)
         teap = backsp(5,iind)
         slope = (den(2,k)- curnt(2)) / (den(1,k)- curnt(1))
         ccint = (yc - curnt(1))*slope + curnt(2)
         err = dmax1(err,2.D0* abs(cc - ccint)/(abs(cc + ccint) + zero))
         slope = (den(3,k)- curnt(3)) / (den(1,k)- curnt(1))
         rhoint = (yc - curnt(1))*slope + curnt(3)
         err = dmax1(err,2.D0* abs(rho - rhoint)/(abs(rho + rhoint) + zero))
         wccal = cc / rhoint
         w1 = curnt(2)/curnt(3)
         w2 = den(2,k)/den(3,k)
         slope = (den(4,k)- curnt(4)) / (w2 - w1)
         entint = (wccal - w1)*slope + curnt(4)
         err = dmax1(err,2.D0* abs(ennix - entint)/(abs(ennix + entint) + zero))
         slope = (den(5,k) - curnt(5)) / (w2 - w1)
         temint = (wccal - w1)*slope + curnt(5)
         err = dmax1(err,2.D0* abs(teap - temint)/(abs(teap + temint) + zero))
180      continue
c
      if(err .le. tcrit) then
         if(ind .eq. iback) soto 200
         ind = ind + 1
         do 190 JJ=1,5
190           backsp(JJ,ind) = curnt(JJ)
              soto 300
         endif

```

```

c
c   record a point in DEN
c
c
200  k = k+1
      if(k.ge. isen) call trap(28)
      do 250 jj=1,5
        den(jj,k) = backsp(jj,ind)
250  backsp(jj,1) = curnt(jj)
      ind = 1
c
300  continue
c
      k = k+1
      if(k.ge. isen) call trap(28)
      if(wc.co. 1.000) then
        den(1,k) = 1.000           ! wc
        den(2,k) = sas_rhoe       ! cc
        den(3,k) = sas_rhoe       ! rhoe
        den(4,k) = enthalpy       ! enthalpy
        den(5,k) = sas_temp       ! temp
      else
        call tprop(2,wc,wa,enthalpy,den(1,k),wa,wa,den(5,k),den(3,k),cp)
        den(2,k) = wc*den(3,k)    ! cc
        den(4,k) = enthalpy
      endif
      den(1,k+1) = 2.             ! .st. 1. end-of-record indicator
c
      return
end
c
c
c
c
c   subroutine addheat(cc,dh,rho,temp,cp)
c
c
c   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
c
c   include 'sys$desadis:DEGADIS1.dec'
c
c   parameter (tfrac=0.618, tfraci=1.-tfrac,
c1         rcrit=0.005, acrit=1., zero=1.e-20)
c
c   common
c   $/GEN2/ DEN(5,isen)
c   $/com_3prop/ sas_w, sas_temp, sas_rhoe, sas_cpk, sas_cpe,
c   $ sas_ufl, sas_lfl, sas_zsp, sas_name
c   $/comata/ istab, tamb, pamb, humid, isofl, tsurf, ihtfl, htco, iutfl, wtco
c
c   character*3 sas_name

```



```

C
c*** data for air/water sys
C
  data waa/28.96/      ! molecular weight of air
  data waw/18./        ! molecular weight of water
  data rho_water/1000./ ! liquid water density [=] kg/m**3
  data cpa/1.0063e3/    ! heat capacity of air [=] J/kg/K
  data cpw/1865./       ! heat capacity of water vapor [=] J/kg/K
  data dhvap/2.5023e6/  ! latent heat of vap [=] J/kg water
  data dhfus/0.33e6/    ! latent heat of fus [=] J/kg water
C
  logical rev
C
C
  vapor_p(txxx) = 6.0298e-3* exp(5407. *(1./273.15- 1./txxx))
  sat_hum(p_tot,p_vp) = 0.622* p_vp/ (p_tot- p_vp) ! kg w/kg BDA
  rho_wair(p_tot,humxx) =
  1      (.00283+ .00456*humxx)/p_tot/(1.+humxx) ! m**3/kg /K
C
C
  cp = cpa
  IF(isof1.eq.1 .or. ihtf1.eq.0) return ! adiabatic mixing is valid
  rhoa = den(3,1)
C
  call adibat(0,wc,wa,yc,ya,cc,rho,wa,enthalpy,amt)
  temp = amt
  if(dh.eq. 0.) return
  enthalpy = enthalpy + dh
C
C
  if(dh.lt. 0.) return ! catch colder surface temperatures
C
C
  if(enthalpy.gt. 0.) then
    temp = tamb
    goto 400
  endif
C
C
C
100 continue
  tmin = amt ! adiabatic mixing temp
  tmin0 = tmin
  tmax = dmax1(tsurf,tamb)
  tmax0 = tmax
  temp = (tmin+tmax)/2.
C
  do 300 j=1,35
    suess = enthal(wc,wa,temp)
    dif = enthalpy - suess
  300
13 -- sys$desadis:TPROP.FOR

```

```

sum = (abs(enthalpy) + abs(guess))/2. + zero
if(abs(dif)/sum.le.rcrit .or. abs(dif).le.acrit) goto 400
c
if(dif.lt. 0.) then
    if(rev) tmax = temp
    if(.not.rev) tmin = temp
    temp = tmin + (tmax-tmin) * tfrac
else
    if(rev) tmin = temp
    if(.not.rev) tmax = temp
    temp = tmin + (tmax-tmin) * tfrac1
endif
c
300 continue
rev = .not.rev
if(rev) goto 100
c
write(lunlog,8050) wc,wa,enthalpy,guess,temp
c8050 format(' ADDHEAT? wc: ',1ps12.5,1x,
c      1      'wa: ',1ps12.5,1x,'enthalpy: ',1ps12.5,/,
c      1      'guess: ',1ps13.5,' temp: ',1ps13.5)
if(temp.lt. sat) call trap(17)
elow = enthal(wc,wa,tmin0)
if(enthalpy.lt. elow) then      ! catch out of bounds numbers
    temp = tmin0
    enthalpy = elow
    goto 400
endif
elow = enthal(wc,wa,tmax0)
if(enthalpy.gt. elow) then
    temp = tmax0
    enthalpy = elow
    goto 400
endif
c
call trap(17)
c
c
100 continue      ! density calculation
vp = vapor_P(temp)
sat = dmini( sat_hum(pamb,vp), humid)
rho = 1./((temp*propu_air(pamb,sat)*wa*(1.+sat)
1      + wc*temp/gas_temp/gas_rhoe + (ww-wa*sat)/rho_water)
if(temp.ne.sat) cp = dmax1(dh/(temp-sat),cpa)
if(temp.lt.sat) stop ' ADDHEAT? wrong way.'
c
c
c
return
end
####

```

```

C.....
C
C   FILE NAME TRANS1 -- FOR USE IN DEGADIS1
C
C.....
C
C   SUBROUTINE TRANS(FILE)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       include 'sys$degadis:DEGADIS1.dec'
C
C   BLOCK COMMON
C
C   COMMON
$/GEN3/ rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
$ srcwa(2,max1),srcenth(2,max1)
$/TITL/TITLE
$/GEN1/ ET(2,isen),R1T(2,isen)
$/GEN2/ DEN(5,isen)
$/ITI/T1,TINP,TSRC,TOBS,TSRT
$/ERROR/STPIN,ERBND,STPMX,WTRG,WTta,WTya,wtsc,wtcb,wtab,wtuh,XLI,
$ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srcct,
$ htcut,ERNOBL,NOBLat,crfser,epsilon
$/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
$/com_sprop/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpe,
$ sas_ufl,sas_lfl,sas_zsp,sas_name
$/comata/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iwtfl,wtco
$/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
$/com_ss/ ess,slen,suid,outcc,outcz,outb,outl,swcl,sual,senl,srhl
$/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
$/com_sisx/ sisx_coeff,sisx_pow,sisx_min_dist,sisx_flg
$/com_enthal/ h_masrte,H_jirrte,H_watrte
$/NEND/ POUNDN,POUND
$/ALP/ ALPHA,alpha1
$/phicom/ iphifl,dellay
$/srd_con/ ce, delrhomin
$/COM_SURF/ HTCUTS
C
C   character*80 TITLE(4)
C
C   character*4 pound
C   character*24 TSRC,TINP,TOBS,TSRT
C   character*3 sas_name
C
C   REAL*8 ML,K
C   LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C   character*(*) file
C
C   ! -- sys$degadis:TRANS1.FOR

```

```

      OPEN(UNIT=8,NAME=FILE,TYPE='NEW',
    1   $   carriasecontrol='list',
    2   $   recordtype='variable')
C
      WRITE(8,1000) (TITLE(I),I=1,4)
1000  FORMAT(A80)
C
      DO 100 I=1,isen
100  IF(ET(1,I).EQ.POUNDN .AND. ET(2,I).EQ.POUNDN) GO TO 105
      stop ' POUND WAS NOT DETECTED '
105  NP = I - 1
      WRITE(8,1040) NP
      DO 110 I=1,NP
110  WRITE(8,1030) ET(1,I),ET(2,I),R1T(2,I)
C
      DO 120 I=1,isen
120  IF(DEN(1,I) .st. 1.) GOTO 125
      DO 122 I=1,isen
122  WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
      stop ' density function blew the loop '
125  NP = I - 1
      WRITE(8,1040) NP
      DO 130 I=1,NP
130  WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
C
      DO 140 I=1,max1
C      cc = srcwc(2,i)*srcden(2,i)
C      if(cc.lt. cclo) then
C          fcc = 0.
C          do ii=i+1,max1
C              fcc = amax1(srcwc(2,ii)*srcden(2,ii),fcc)
C          enddo
C          if(fcc.ge. cc) goto 140
C          np = i
C          tend = srcwc(1,i)
C          goto 146
C      endif
140  IF(rads(1,I).EQ.POUNDN .AND. rads(2,I).EQ.POUNDN) GO TO 145
      stop ' POUND WAS NOT DETECTED '
145  NP = I - 1
146  WRITE(8,1040) NP
      DO 150 I=1,NP
150  WRITE(8,1060) rads(1,i),rads(2,i),estr(2,i),srcden(2,i),srcwc(2,i)
      1      ,srcwc(2,i),srcden(2,i)
C
1020  format(1x,i4,1x,1p14.7)
1030  format(3(1x,1p14.7))
1040  format(1x,i4)
1050  format(2(a24,1x))
1060  format(7(1x,1p14.7))
1070  format(1x,1p14.7)

2 -- sysidesadis:TRANS1.FOR

```

```

1080 format(a3)
C
WRITE(8,1050) TINP,TSRC
write(8,1050) TOBS,TSRT
WRITE(8,1060) UO,ZO,ZR,ML,USTAR
write(8,1060) K,G,RHOE,RHOA,DELTA
write(8,1030) BETA,GAMMAF,CcLOW
WRITE(8,1060) RM,SZM,EMAX,RMAX,TSC1
write(8,1030) ALEPH,TEND
WRITE(8,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
WRITE(8,1070) ALPHA
write(8,1080) sas_name
write(8,1030) sas_mu,sas_temp,sas_rhoe
write(8,1030) sas_cpk,sas_cpf
write(8,1030) sas_ufl,sas_lfl,sas_zsp
write(8,1040) istab
write(8,1030) tamb,pamb,humid
write(8,1020) isofl,tsurf
write(8,1020) ihtfl,htco
write(8,1020) iwtfl,wtco
write(8,1030) sixx_coeff,sixx_pow,sixx_min_dist
C
if(check4) then
    write(8,1030) ess,slen,swid
    write(8,1060) outcc,outcz,outb,outl
    write(8,1060) swcl,swal,senl,srhl
    end if
C
write(8,1020) iphifl,dellay
C
if(isofl.eq. 0) write(8,1030) H_nasrte
C
WRITE(8,1030) HTCUTS, ce, delrhoain
C
CLOSE(UNIT=8)
C
RETURN
END
****

```

```

C.....
C
C   FILE NAME TRANS2 -- USE WITH DEGADIS2
C
C   SUBROUTINE TRANS(FILE)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       include 'sys$desadis:DEGADIS2.dec'
C
C       COMMON
C       $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
C       $/GEN2/ DEN(5,isen)
C       $/ITI/ T1,TINP,TSRC,TOBS,TSRT
C       $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C       $/COM_PROP/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C       $ sas_ufl,sas_lfl,sas_zsp,sas_name
C       $/COMATA/ istab,tamb,pamb, humid, isofl,tsurf,ihtfl,htco,iutfl,wtco
C       $/PARMSC/ RM,SM,EMAX,RMAX,TSC1,ALEPH,TEND
C       $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C       $/COM_six/ six_coeff,six_pow,six_min_dist,six_flg
C       $/nend/ poundn,pound
C       $/ALP/ ALPHA,alpha1
C       $/CNOBS/ NOBS
C
C       character*3 sas_name
C       character*80 TITLE(4)
C       character*24 TINP,TSRC,TOBS,TSRT
C       character*(*) file
C
C       REAL*8 K,ML
C       LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C       OPEN(UNIT=9,NAME=FILE,TYPE='NEW',
C       $ carriagecontrol='list',
C       $ recordtype='variable')
C
C       WRITE(9,1040) NOBS
C       DO 125 I=1,NOBS
125 WRITE(9,1010) NREC(I,1),NREC(I,2),T0(I),XV(I)
C
C       DO 140 I=1,isen
140 IF(DEN(1,I).gt. 1.) GOTO 145
C       stop ' density function error in TRANS'
145 NP = I - 1
C       WRITE(9,1040) NP
C       DO 150 I=1,NP
150 WRITE(9,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
C
C       WRITE(9,1060) U0,Z0,ZR,ML,USTAR
C
1 -- sys$desadis:TRANS2.FOR

```

C-170

```
      write(9,1060) K,G,RHOE,RHOA,DELTA
      write(9,1030) BETA,GAMMAF,CcLOW
C
      WRITE(9,1050) Tinp,TSRC
      write(9,1050) TOBS,TSRT
C
      WRITE(9,1060) RM,SZM,EMAX,RMAX,TSC1
      write(9,1020) ALEPH,TEND
C
      write(9,1080) sas_name
      write(9,1030) sas_sur,sas_temp,sas_rhoe
      write(9,1020) sas_cpk,sas_cpf
      write(9,1030) sas_ufl,sas_lfl,sas_zsp
      write(9,1040) istab
      write(9,1030) tamb,psab, humid
      write(9,1025) isofl,tsurf
      write(9,1025) ihtfl,htco
      write(9,1025) iutfl,utco
      write(9,1030) sixx_coeff,sixx_pow,sixx_min_dist
C
      WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      WRITE(9,1070) ALPHA
C
1010 format(1x,i8,1x,i8,2(1x,1p14.7))
1020 format(2(1x,1p14.7))
1025 format(1x,i4,1x,1p14.7)
1030 format(3(1x,1p14.7))
1040 format(1x,i4)
1050 format(2(a24,1x))
1060 format(5(1x,1p14.7))
1070 format(1x,1p14.7)
1090 format(a3,1x)
C
      CLOSE(UNIT=9)
      RETURN
      END
####
```

```

C.....
C
C   FILE NAME TRANS2 -- USE WITH SDEGADIS2
C
C.....
C
C   SUBROUTINE TRANS(FILE)
C
C
C
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   COMMON
C   $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C   $/COM_PROP/ sas_aw,sas_temp,sas_rhoe,sas_cpk,sas_cpf,
C   $ sas_ufl,sas_lfl,sas_zsp,sas_name
C   $/COMETA/ istab,tamb,psab, humid, isofl,tsurf, ihtfl,htco, iwtfl,wtco
C   $/ITI/ t1,TINP,TSRC,TOBS
C   $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C   $/ALP/ALPHA,alpha1
C
C   character*24 TSRC,TINP,TOBS
C   character*3 sas_name
C   character*(*) file
C
C   REAL*8 K,ML
C   LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
C   OPEN(UNIT=9,NAME=FILE,TYPE='NEW')
C
C   WRITE(9,1060) U0,Z0,ZR,ML,USTAR
C   write(9,1060) K,G,RHOE,RHOA,DELTA
C   write(9,1030) BETA,GAMMAF,CcLOW
C
C   WRITE(9,1050) TINP,TSRC
C   write(9,1050) TOBS
C
C   write(9,1080) sas_name
C   write(9,1030) sas_aw,sas_temp,sas_rhoe
C   write(9,1020) sas_cpk,sas_cpf
C   write(9,1030) sas_ufl,sas_lfl,sas_zsp
C
C   write(9,1040) istab
C   write(9,1030) tamb,psab, humid
C   write(9,1025) isofl,tsurf
C   write(9,1025) ihtfl,htco
C   write(9,1025) iwtfl,wtco
C
C   WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
1 -- sysdegadis:TRANS2SS.FOR

```



C-172

```
WRITE(9,1070) ALPHA
```

```
C
```

```
CLOSE(UNIT=9)
```

```
C
```

```
1020 format(2(1x,1p14.7))
```

```
1025 format(1x,i4,1x,1p14.7)
```

```
1030 format(3(1x,1p14.7))
```

```
1040 format(1x,i4)
```

```
1050 format(2(a24,1x))
```

```
1060 format(5(1x,1p14.7))
```

```
1070 format(1x,1p14.7)
```

```
1080 format(a3,1x)
```

```
C
```

```
RETURN
```

```
END
```

```
****
```

```

C      FILE NAME TRANS3 FOR USE WITH DEGADIS3
C
C.....
C
C      SUBROUTINE TRANS(OPNRUP)
C
C      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C      include 'sys$desadis:DEGADIS3.dec/list'
C
C      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
$          Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
$          Tsamaa(maxnob,maxnt),Ttemp(maxnob,maxnt),
$          TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
$          TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
$ /SORTIN/TIM(maxnt),NTIM,ISTRT
$ /ITI/TI,TINP,TSRC,TOBS,TSRT
C
C      character*24 tinp,tsrc,tobs,tsrt
C
C      character*(*) OPNRUP
C
C      TO = TIM(ISTRT)
C      DT = TIM(ISTRT+1) - TIM(ISTRT)
C
C      OPEN(UNIT=9,NAME=OPNRUP,TYPE='NEW',
$          carriagecontrol='list',recordtype='variable')
C
C
C      DO 110 I = ISTRT,NTIM
C          II = I - ISTRT + 1
C      110 KSUB(II) = KSUB(I)
C          NTIM = NTIM - ISTRT + 1
C          IF(NTIM.EQ. maxnt) GO TO 120
C          II = NTIM + 1
C          DO 115 I=II,maxnt
C      115 KSUB(I) = 0
C
C      120 CONTINUE
C
C      WRITE(9,*) TO,DT,NTIM
C      WRITE(9,*) KSUB
C
C      CLOSE(UNIT=9)
C
C      RETURN
C      END
####

```

```

C.....
C
C   SUBROUTINE TRAP -- DIAGNOSTICS
C
C   SUBROUTINE trap(N,N1)
C
C       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C       include 'sys$desadis:DEGADIS1.dec'
C
C       COMMON /ITI/T1,TINP,TSRC,TOBS,TSRT
C
C       real*4 tt1
C
C       character*24 TINP,TSRC,TOBS,TSRT
C
C       character*24 tt
C
C       WRITE(lunlos,1100)
C       WRITE(lunlos,1110)
C       write(lunlos,1115) n
C
C       IF(N .EQ. 1 ) WRITE(lunlos,2010) N1
C       IF(N .EQ. 2 ) WRITE(lunlos,2020)
C       IF(N .EQ. 3 ) WRITE(lunlos,2030) N1
C       IF(N .EQ. 4 ) WRITE(lunlos,2040)
C       IF(N .EQ. 5 ) WRITE(lunlos,2050)
C       IF(N .EQ. 6 ) WRITE(lunlos,2060)
C       IF(N .EQ. 7 ) WRITE(lunlos,2070)
C       IF(N .EQ. 8 ) WRITE(lunlos,2080) N1
C       IF(N .EQ. 9 ) WRITE(lunlos,2090) N1
C       IF(N .EQ. 10) WRITE(lunlos,2100) N1
C       IF(N .EQ. 11) WRITE(lunlos,2110)
C       IF(N .EQ. 12) WRITE(lunlos,2120)
C       IF(N .EQ. 13) WRITE(lunlos,2130)
C       IF(N .EQ. 14) WRITE(lunlos,2140)
C       IF(N .EQ. 15) WRITE(lunlos,2150)
C       IF(N .EQ. 16) WRITE(lunlos,2160)
C       IF(N .EQ. 17) WRITE(lunlos,2170)
C       IF(N .EQ. 18) WRITE(lunlos,2180) N1
C       IF(N .EQ. 19) WRITE(lunlos,2190) N1
C       IF(N .EQ. 20) WRITE(lunlos,2200)
C       IF(N .EQ. 21) WRITE(lunlos,2210)
C       IF(N .EQ. 22) WRITE(lunlos,2220)
C       IF(N .EQ. 23) WRITE(lunlos,2230)
C       IF(N .EQ. 24) WRITE(lunlos,2240)
C       IF(N .EQ. 25) WRITE(lunlos,2250)
C       IF(N .EQ. 26) WRITE(lunlos,2260)
C       IF(N .EQ. 27) WRITE(lunlos,2270)
C       IF(N .EQ. 28) WRITE(lunlos,2280)

```

1 -- sys\$desadis:TRAP.FOR

```

IF(N .EQ. 29) WRITE(lunlos,2290)
IF(N .EQ. 30) WRITE(lunlos,2300)
IF(N .EQ. 31) WRITE(lunlos,2310)
IF(N .EQ. 32) WRITE(lunlos,2320)
IF(N .EQ. 33) WRITE(lunlos,2330)

```

C

```

1100 FORMAT(5X,'The best laid plans of mice and men...')
1110 FORMAT(5X,'You have entered a TRAP -- the land of no RETURN.')
1115 format(' Code: ',i4)
2010 FORMAT(5X,'DEGADIS1? SOURCE INTEGRATION HAS RETURNED IHLF=',I3)
2020 FORMAT(5X,'Reserved')
2030 format(5X,'SZF? Local integration failed; IHLF=',I3)
2040 format(5X,'SURFACE? Negative Q RTE for positive DELTA_T')
2050 FORMAT(5X,'CRFG? MORE POINTS FOR GEN3 WERE NEEDED')
2060 FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TUPF FAILED')
2070 FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TDMF FAILED')
2080 FORMAT(5X,'SSSUP? OBSERVER INTEGRATION FAILED, IHLF=',I3)
2090 FORMAT(5X,'SSSUP/SDEGADIS2? PSEUDO-STEADY INTEG FAILED, IHLF=',I3)
2100 FORMAT(5X,'SSSUP/SDEGADIS2? GAUSSIAN INTEGRATION FAIL, IHLF=',I3)
2110 FORMAT(5X,'SSSUP/SDEGADIS2? TOTAL No. OF RECORDS EXCEED 120000')
2120 FORMAT(5X,'Reserved')
2130 FORMAT(5X,'Reserved')
2140 FORMAT(5X,'Reserved')
2150 FORMAT(5X,'Reserved')
2160 FORMAT(5X,'PSSOUT/PSSOUTSS? PSS STARTED WITH B<0.')
2170 format(5X,'TPROP/ADDHEAT? Enthalpy out of bounds')
2180 FORMAT(5X,'ALPH? ALPHA INTEGRATION FAILED, IHLF=',I3)
2190 FORMAT(5X,'ALPH? RTMI HAS FAILED TO LOCATE ALPHA IERR: ',i4)
2200 format(5X,'ESTRT? Premature EOF in RUN_NAME.ER1 or RUN_NAME.ER2.')
2210 FORMAT(5X,'ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed')
2220 format(5X,'ESTRT1? The parameter file RUN_NAME.ER1 wasnot found.')
2230 format(5X,'SORTS1? Fewer than 3 points sorted for any time.')
2240 format(5X,'TPROP? Trial and error loop compromised')
2250 format(5X,'TPROP? Isothermal density loop compromised')
2260 format(5X,'TPROP? Invalid entry flag in ADIABAT')
2270 format(5X,'Reserved')
2280 format(5X,'TPROP? IGEN request too large in SETDEN')
2290 format(5X,'PHIF? flag out of bounds')
2300 format(5X,'SSSUP/SDEGADIS2? concentration greater than RHOE')
2310 format(5X,'SSSUP? concentration greater than RHOE')
2320 format(5X,'PSS? Sz convergence failure.')
2330 format(5X,'SSG? Sz convergence failure.')

```

C

```

CLOSE(UNIT=9)

```

C

```

CALL TRANS('trap.DBG')

```

C

```

istat = lib$date_time(TT)
tt1 = t1
ttime = secnds(tt1)/60.

```

C

```

2 -- sys$desadis:TRAP.FOR

```

C-176

```
140 WRITE(lunlog,3000) TT
    WRITE(lunlog,3010) Ttime
3000 FORMAT(1X,' -- ENDing AT ',A24)
3010 FORMAT(5X,' ***** ELAPSED TIME ***** ',1P13.5,' MIN ')
C
    CALL EXIT
    END
****
```

3 -- sys\$desadis:TRAP.FOR

C-177

```
C.....  
C  
C   FUNCTION TO CALCULATE A SPECIFIED TIME  
C  
C   FUNCTION TS(T01,DIST)  
C  
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )  
  
C   COMMON  
C   $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND  
C   $/ALP/ALPHA,alpha1  
  
C   TS = T01 + (DIST+RMAX)**(1./ALPHA1) /ALEPH  
C  
C   RETURN  
C   END  
****
```

```

C.....
C
C   OBSERVER TRIAL AND ERROR FUNCTIONS
C   ---- TUPF ---- TDNF ----
C
C   Modified 4 Nov 85 to account for more general forms of the
C   Gas Radius as a function of time.
C
C   FUNCTION TUPF(T01)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   include 'sys$desadis:DEGADIS2.dec'
C
C   COMMON
C   $/GEN3/ rads(2,max1),astr(2,max1),srcden(2,max1),srcwc(2,max1),
C   $ srcwa(2,max1),srcenth(2,max1)
C   $/ERROR/SYOER,ERRO,SZOER,WTALO,WTQOO,WTSDO,ERRP,SMXP,
C   $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WDHNG
C   $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C   $/ALP/ALPHA,alpha1
C
C   LOGICAL REV, LAST, pflag
C   REV = .FALSE.
C   LAST = .FALSE.
C   pflag = .false.
C
C   stop ' use tupf.old as the source for this routine'
C   TMAXO = RMAX*(1./ALPHA1)/ALEPH + T01
C   TMINO = TOL
130  TMIN = TMINO
C   TMAX = TMAXO
C   IF(T01 .LT. 0.) TMIN = 0.
C   T1 = (TMAX + TMIN)/2.
C
C   DO 100 I = 1,100
C   II = 0
110  XG = -AFGEN(RADG,T1,'tupf')
C   XO = XIT(T1,T01)
C   IF(XO .LT. 0.) GO TO 120
C   T1 = (T1+TMIN)/2.
C   II = II + 1
C   if(.pflag) write(6,5020) t1,t01,xs,xo
5020 format(' t1:',1ps13.5,' t01:',1ps13.5,' xs:',1ps13.5,
C   1 ' xo:',1ps13.5)
C   IF(II.EQ. 20) GOTO 101
C   GO TO 110
C
C   120 CONTINUE
C   DIF = XO - XG
C
1  -- sys$desadis:TUPF.FOR

```

```

      sum = (xo + xs)/2.
      IF(ABS(DIF)/ABS(sum) .LT. ERTUPF) GO TO 1000
      if(pflag) write(6,5040) tmin,tmax,tl,xo,xs
5040  format(' tmin:',1ps13.5, ' tmax:',1ps13.5, ' tl:',1ps13.5,
1 ' xo:',1ps13.5, ' xs:',1ps13.5)
      IF(REV) GO TO 140
      IF(DIF .GT. 0.) TMAX = T1
      IF(DIF .LT. 0.) THIN = T1
      GO TO 100
C
140  IF(DIF .LT. 0.) TMAX = T1
      IF(DIF .GT. 0.) THIN = T1
C
100  T1 = THIN + 0.5*(TMAX-THIN)
C
101  IF(.NOT. REV) GO TO 150
      IF(LAST) then
        if(pflag) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
4000  format(' rm:',1ps13.5, ' szm:',1ps13.5, ' emax:',1ps13.5,/,
1 ' rmax:',1ps13.5, ' tsc1:',1ps13.5, ' aleph:',1ps13.5,/,
2 ' tend:',1ps13.5, ' alpha:',1ps13.5)
        if(pflag) write(6,4010) tmax0,tmin0
4010  format(' tmax0:',1ps13.5, ' tmin0:',1ps13.5)
        CALL trap(6)
      endif
      TMAX0 = 1.1*TMAX0
      TMINO = 0.92*TMINO
      REV = .FALSE.
      LAST=.TRUE.
      pflag=.true.
150  CONTINUE
      t1 = TL+.01
      T2 = TL-.01
      XG1 = AFGEN(RADG,T1,'tupf')
      xs2 = AFGEN(RADG,T2,'tupf')
      dif = abs(xs1-xs2)
      if(dif.gt. 100. .AND. (X0.GE.XG2 .AND. X0.LE.XG1)) then
        tupf = t2
        ! JUMP FROM BLANKET TO NOBLANKET
        RETURN
      endif
      REV = .TRUE.
      GO TO 130
C
1000 TUPF = T1
      IF(REV) WRITE(1unlos,1100)
1100  FORMAT(1X,'?TUPF? -- REV WAS TRUE -- ',49X,'Z')
      RETURN
      END
C
      FUNCTION TDNF(T01)
C
2 -- sys$desadis:TUPF.FOR

```



```

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$desadis:DEGADIS2.dec'

C
      COMMON
      $/GEN3/ rads(2,max1),estr(2,max1),srcden(2,max1),srcwc(2,max1),
      $ srcwa(2,max1),srcenth(2,max1)
      $/ERROR/SYOER,ERRO,SZOER,WTAI0,WTQ00,WT SZ0,ERRP,SMXP,
      $ WTSZP,WT SYP,WTBEP,WT DH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WT DHG
      $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
      $/ALP/ALPHA,alpha1

C
      LOGICAL REV, LAST, rflag
      REV = .FALSE.
      LAST = .FALSE.
      rflag = .false.

C
      TMINO = RMAX**((1./ALPHA1)/ALEPH + T01
      TMAX0 = (2.*RMAX)**((1./ALPHA1)/ALEPH + T01

C
      100 TMIN = TMINO
      TMAX = TMAX0
      IF(TM IN .LT. 0.) TM IN = 0.
      T = (TMAX + TM IN)/2.

C
      DO 110 I = 1,100
      II = 0
      120 XG = AFGEN(RADG,T,'tdnf')
      XO = XIT(T,T01)

C
      IF(XO .GT. 0.) GO TO 130
      T = (TMAX + T)/2.
      II = II + 1
      if(rflag) write(6,5020) t,t01,xs,xo
      5020 format(' t:',1ps13.5,' t01:',1ps13.5,' xs:',1ps13.5,
      1 ' xo:',1ps13.5)
      IF(II.EQ. 20) GOTO 111
      GO TO 120
      130 CONTINUE

C
      DIF = XO - XG
      sum = xo+xs
      IF(ABS(DIF)/abs(sum) .LT. ERTDNF) GO TO 1000
      if(rflag) write(6,5040) tain,tmax,t,xo,xs
      5040 format(' tain:',1ps13.5,' tmax:',1ps13.5,' t:',1ps13.5,
      1 ' xo:',1ps13.5,' xs:',1ps13.5)

C
      IF(REV) GO TO 140
      IF(DIF .GT. 0.) TMAX = T
      IF(DIF .LT. 0.) TM IN = T

      3 -- sys$desadis:TUPF.FOR

```

```

      GO TO 110
C
140 IF(DIF .LT. 0.) TMAX = T
    IF(DIF .GT. 0.) TMIN = T
C
110 T = TMIN + 0.5*(TMAX - TMIN)
C
111 IF(.NOT. REV) GO TO 150
    IF(LAST) then
      if(.nflas) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
4000 format(' rm:',1p13.5,' szm:',1p13.5,' emax:',1p13.5,/,
1 ' rmax:',1p13.5,' tsc1:',1p13.5,' aleph:',1p13.5,/,
2 ' tend:',1p13.5,' alpha:',1p13.5)
      if(.nflas) write(6,4010) tmax0,tmin0
4010 format(' tmax0:',1p13.5,' tmin0:',1p13.5)
      CALL trap(7)
    endif
      TMAX0 = 1.1*TMAX0
      TMIN0 = 0.92*TMIN0
      REV = .FALSE.
      LAST=.TRUE.
      nflas = .true.
      GOTO 100
C
150 CONTINUE
    t1 = T+.01
    T2 = T-.01
    XG1 = AFGEN(RADG,T1,'tdnf')
    xg2 = AFGEN(RADG,T2,'tdnf')
    dif = abs(xg1-xg2)
    if(dif.gt. 100. .AND. (X0.LE.XG2 .AND. X0.GE.XG1)) then
      tdnf = t2
      ! JUMP FROM BLANKET TO NOBLANKET
      RETURN
    endif
    REV = .TRUE.
    GO TO 100
1000 TDNF = T
    IF(REV) WRITE(1unlog,1100)
1100 FORMAT(5X,'?TDNF? -- REV WAS TRUE',49X,'Z')
    RETURN
  END
####

```

```

C.....
C
C   FUNCTIONS ASSOCIATED WITH THE OBSERVER CALCULATIONS
C
C.....
C   FUNCTION TO RETURN OBSERVER VELOCITY AS A FUNCTION OF TIME
C
C   FUNCTION UIT(T,T01)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   COMMON
C   $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C   $/ALP/ALPHA,alpha1
C
C   UIT = ALPHA1 * ALEPH**ALPHA1 *(T-T01)**ALPHA
C
C   RETURN
C   END
C
C.....
C   FUNCTION TO RETURN POSITION AS A FUNCTION OF TIME AND TO
C
C   FUNCTION XIT(T1,T01)
C
C   Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
C   COMMON
C   $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
C   $/ALP/ALPHA,alpha1
C
C   xit = -rmax
C   arg = t1-t01
C   if(arg .le. 0.) return
C   XIT = (ALEPH*(T1 - T01))**ALPHA1 - RMAX
C
C   RETURN
C   END
C
C.....
C   FUNC TO RETURN A VALUE OF TO BASED ON A POSITION AND TIME
C
C   -- sys$desadis:UIT.FOR

```

```
FUNCTION TOOB(X,T)
```

```
C
```

```
Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
```

```
COMMON
```

```
$/PARMSC/RH, SZH, EMAX, RMAX, TSC1, ALEPH, TEND
```

```
$/ALP/ALPHA, alpha1
```

```
C
```

```
ARG = 0.
```

```
CHECK = ABS((ABS(X)-ABS(RMAX)))/(ABS(X)+ABS(RMAX))
```

```
IF(CHECK .GT. 0.001) ARG = (X + RMAX)**(1./ALPHA1)/ALEPH
```

```
TOOB = T - ARG
```

```
RETURN
```

```
END
```

```
****
```

## APPENDIX D

## ERROR MESSAGES

To assist the user in determining the source of any problems, a diagnostic procedure has been included in DEGADIS. The subroutine TRAP is meant to cause an orderly termination of the program for many detected errors. It performs two basic functions: TRAP displays an error code and a single line diagnostic message giving the reason for premature termination, and TRAP forces an output of the COMMON area data sets to the file TRAP.DBG.

The first three lines sent to the execution log (default-TERMINAL) include the TRAP introductory lines and the error code number:

```
The best laid plans of mice and men . . .
You have entered a TRAP--THE LAND OF NO RETURN
CODE:  NN
```

where NN represents the code of the error message which follows in the log. The error message begins with the name of the calling routine.

The following is a list of the error codes, error messages, and suggested actions for each problem.

Code: 1

DEGADIS1? Source integration has returned IHLF-NN

Action: This error occurs during integration of the equations describing the gas source. NN is an error code returned by the integration package RKGST.

When NN-11, more than 10 bisections of the initial increment of the independent variable were necessary to take an integration step within the specified error. Reduce the initial step size of the independent variable (ER1 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ER1 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (ER1 file).

When NN-12, the initial increment of the independent variable is 0. Correct the ER1 file and execute the program again.

When NN-13, the initial increment of the independent variable is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. Correct the ER1 file and execute the program again.

Code: 2

Reserved

Action: Not applicable

Code: 3

SZF? Local integration failed; IHLF-NN

Action: This error occurs during estimation of the value of  $S_2$  over the source when no gas blanket is present. See Code: 1 for appropriate actions.

Code: 4

SURFACE? Negative QRTE for positive DELTA\_T

Action: Diagnostic message indicating an error in estimation of the heat capacity.

Code: 5

CRFG? More points for GEN3 were needed

Action: The COMMON area /GEN3/ stores representative values of the calculated source parameters. If this condition occurs, relax the CRFG error criteria in the ER2 file. If this is a common problem, the length of the /GEN3/ vectors can be increased by changing the value of MAXL in DEGADIS1.DEC and reinstalling DEGADIS.

Code: 6

TUPF? Observer calculations--TUPF failed

Action: The trial and error search associated with finding the upwind edge of the gas source for an observer failed. Often this problem can be avoided by adding one or two additional observers to the present number of observers (which moves the solution of the trial and error). Another possibility is to increase the error criteria for this function in the ER2 file.

Code: 7

TUPF? Observer calculations--TDNF failed

Action: The trial and error search associated with finding the downwind edge of the gas source for an observer failed. Often this problem can be avoided by adding one or two additional observers to the present number of observers (which moves the solution of the trial and error). Another possibility is to increase the error criteria for this function in the ER2 file.

Code: 8

SSSUP? Observer integration failed; IHLF=NN

Action: This error occurs during integration of the four differential equations which average the source for each observer. See Code: 1 for appropriate actions.

Code: 9

SSSUP/SDEGADIS2? Pseudo-Steady integration failed, IHLF=NN

Action: This error occurs during integration of the four differential equations describing the portion of the downwind calculation when  $b > 0$ . The routine calling TRAP is SSSUP if a transient simulation is being executed; if a steady state simulation is being executed, the calling routine is SDEGADIS2. See Code: 1 for appropriate actions.



Code: 10

SSSUP/SDEGADIS2? Gaussian integration failed, IHLF-NN

Action: This error occurs during integration of the differential equations which describe the portion of the downwind calculation when  $b = 0$ . The routine calling TRAP is SSSUP if a transient simulation is being executed; SDEGADIS2 is calling TRAP if a steady state simulation is being executed. See Code: 1 for appropriate actions.

Code: 11

SSSUP/SDEGADIS2? TOTAL NO. OF RECORDS EXCEEDS 120,000

Action: This is an arbitrary stopping point for the process in order to keep a runaway simulation from filling up disk space. Relax the output specifications in the ER2 file in order to generate less output.

Code: 12

Reserved

Action: Not applicable

Code: 13

Reserved

Action: Not applicable

Code: 14

Reserved

Action: Not applicable

Code: 15

Reserved

Action: Not applicable

Code: 16

PSSOUT/PSSOUTSS? PSS started with  $b < 0$

Action: This condition is checked at the beginning of the downwind calculation in order to confirm proper handling of the movement to the Gaussian phase of the downwind calculation. Correct the initial conditions and execute the program again.

Code: 17

TPROP/ADDHEAT? Enthalpy out of bounds

Action: Diagnostic message indicating an enthalpy lower than the adiabatic mixing enthalpy was passed to ADDHEAT.

Code: 18

ALPH? ALPHA integration failed; IHLF=NN

Action: The integration which determines the integral least squares fit for ALPHA has failed. See Code: 1 for appropriate actions. Note that large values of Monin-Obukhov length ( $\lambda > 0(1.0 \text{ m})$ ) in combination with stable atmospheric conditions may cause this integration to fail.

Code: 19

ALPH? RTMI has failed to locate ALPHA; IERR: NN

Action: The search procedure to determine the value of ALPHA has failed. NN is an error code returned by the routine RTMI.

When NN=1, the search for ALPHA failed after a specified number of iterations.

When NN=2, the basic assumption that the function which governs the search for ALPHA changes sign over the specified interval is false.

This error is probably the result of an unusual velocity specification such as small values for the Monin-Obukhov length ( $\lambda < 0(1.0 \text{ m})$ ) or small values for the reference height ( $z_0 < 0(10\lambda)$ ). Also see Code: 18.

Code: 20

ESTRT? Premature EOF in RUN\_NAME.ER1 or RUN\_NAME.ER2.

Action: The portion of the program which reads ER1 and ER2 files encountered an end-of-file before all of the information had been gathered. Confirm these files and execute the program again. If necessary, copy and edit an EXAMPLE file for your application and execute the program again.

Code: 21

ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed

Action: The portion of the program which reads the ER1, ER2, or ER3 file failed to understand a numerical entry. The numbers must appear in columns 11-20 of the line with no alphabetic characters in the field. This does not apply to comment lines which contain an exclamation point (!) in the first column of the line.

Code: 22

ESTRT1? The parameter file RUNNAME.ER1 was not found

Action: The ER1 file was not found for the current simulation (RUNNAME). Copy the EXAMPLE.ER1 file to RUNNAME.ER1 and edit it as necessary. Execute the program again.

Code: 23

SORTS1? Fewer than 3 points sorted for any time

Action: Only one or two simulation points were applicable for the sort times specified. There are two possible causes for this condition: First, sort times specified were either before the simulation had developed significantly, or after the simulation was completed. If additional information is desired at the end of the simulation, restart the simulation and specify a lower concentration of interest in the input step (DEGADISIN). Second, the number of observers specified for the problem was too low to give a good resolution of the downwind concentration field. Increase the number of observers in the ER2 file.

Code: 24

TPROP? Trial and error loop compromised

Action: TPROP estimates the temperature of a mixture based upon the composition and enthalpy of the mixture. Ensure the properties for the diffusing species are entered correctly and execute the simulation again.

Code: 25

Reserved

Action: Not applicable

Code: 26

TPROP? Invalid entry flag in ADIABAT

Action: This is a programming diagnostic. This error should never occur.

Code: 27

Reserved

Action: Not applicable

Code: 28

TPROP? IGEN request too large in SETDEN

Action: The subroutine SETDEN performs a series of adiabatic mixing calculations with a specified gas mixture and ambient air and places the results in the array DEN(5,IGEN). This error indicates more points are needed in DEN than were originally requested. Increase the allocation for DEN by changing the value of IGEN in DEGADISIN.DEC and reinstalling DEGADIS.

Code: 29

PHIF? flag out of bounds

Action: This is programming diagnostic. This error should never occur.

Code: 30

SSSUP/SDEGADIS2? concentration greater than RHOE

Action: If the concentration of the contaminant becomes greater than the pure component density for an isothermal simulation, this error will occur. However, this situation should never occur.

Code: 31

Reserved

Action: Not applicable

Code: 32

PSS? Sz convergence failure

Action: This is a programming diagnostic. This error should never occur.

Code: 33

SSG? Sz convergence failure

Action: This is a programming diagnostic. This error should never occur.

D-12



## APPENDIX E

## PARTIAL LISTING OF PROGRAM VARIABLES

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
AGAIN	LOGICAL			Local communications in SSSUP
ALEPH	REAL			Collection of constants to calculate observer position and velocity
ALPHA	REAL	$\alpha$	n/a	Power law velocity profile power
ALPHA1	REAL	$(1.0 + \alpha)$	n/a	
BETA	REAL	$\beta$	n/a	Lateral similarity power
CCLOW	REAL		kg/m <sup>3</sup>	Lowest mixture concentration of interest
CHECK1	LOGICAL			Unused logical flag
CHECK2	LOGICAL			When true, release type without a liquid source
CHECK3	LOGICAL			Local communications flag used in DEGADIS1

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
CHECK4	LOGICAL			When true, steady state release
CHECK5	LOGICAL			When true, user sets time sort parameters
DELTA	REAL	$\delta$	$m^{1-\beta}$	Lateral similarity coefficient
DEN(1,I)	REAL	$y_c$	mole fraction	Contaminant mole fraction
DEN(2,I)	REAL	$c_c$	$kg/m^3$	Contaminant concentration for the given mole fraction
DEN(3,I)	REAL	$\rho$	$kg/m^3$	Mixture density for the given mole fraction
DEN(4,I)	REAL	$h$	J/kg	Mixture enthalpy for the given mole fraction
DEN(5,I)	REAL	$T$	K	Mixture temperature for the given mole fraction
EMAX	REAL		kg/s	Maximum of secondary source mass evolution rate
ESS	REAL	$E$	kg/s	Steady state release rate
ET(1,I)	REAL	$t$	s	Independent variable time for ordered pairs ET

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
ET(2,I)	REAL	E( $\tau$ )	kg/s	Source mass evolution rate as a function of time characterized by ordered pairs
G	REAL	g	m/s <sup>2</sup>	Acceleration due to gravity
GAMMAF	REAL	$\Gamma(1/(1+\alpha))$	n/a	
GAS_CPK	REAL	q <sub>1</sub>	J/kmol	Constant for contaminant heat capacity
GAS_CPP	REAL	p <sub>1</sub>	n/a	Power for contaminant heat capacity
GAS_LFL	REAL		mole fraction	Lower flammability limit of contaminant
GAS_MW	REAL	MW <sub>C</sub>	kg/kmol	Contaminant molecular weight
GAS_NAME	CHARACTER*3			Name of contaminant
GAS_RHOE	REAL	$\rho_0$	kg/m <sup>3</sup>	Saturated vapor density of contaminant at T <sub>0</sub>
GAS_TEMP	REAL	T <sub>0</sub>	K	Contaminant storage temperature
GAS_UFL	REAL		mole fraction	Upper flammability limit of contaminant

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
GAS_ZSP	REAL		m	Height for estimating flammability contours
GMASS0	REAL		kg	Initial mass of gas over the primary source
HTCO	REAL	$h_0$	$J/m^2 sK$	Constant coefficient when IHTFL=-1
		$v_H$	m/s	LLNL heat transfer velocity when IHTFL=2
HUMID	REAL	kg water/ kg dry air		Ambient absolute humidity
IHTFL	INTEGER			Heat transfer flag: IHTFL=-1 constant coefficient IHTFL=0 no heat transfer IHTFL=1 DEGADIS coefficient IHTFL=2 LLNL coefficient
ISOFL	INTEGER			Isothermal release when ISOFL=1
ISTAB	INTEGER			Pasquill atmospheric stability indicator (ISTAB=1 for A, (ISTAB=2 for B, etc.)
IWTFL	INTEGER			Water transfer flag IWTFL=-1 constant coefficient IWTFL=0 no water transfer IWTFL=1 DEGADIS coefficient

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
K	REAL	k	n/a	von Karman's constant 0.35
LUNLOG	INTEGER			Fortran logical unit number which acts as a simulation log
MAXNOB	INTEGER			Maximum number of observers
ML	REAL	$\lambda$	m	Monin-Obukhov length
NOBS	INTEGER			Number of observers for the pseudo- steady state simulation
NREC(I,1)	INTEGER			Number of records generated in PSSOUT for observer I
NREC(I,2)	INTEGER			Number of records generated in SSGOUT for observer I
PAMB	REAL	p	atm	Ambient pressure
POUND	CHARACTER*4			Character string to signal end of data ('// ')
POUNDN	REAL			Numerical value to signal end of data (-1.E-20)

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
QSTR(1,I)	REAL	t	s	Independent variable time for ordered pairs QSTR
QSTR(2,I)	REAL	$Q_*$	$\text{kg/m}^2 \text{ s}$	Atmospheric takeup rate as a function of time
RADG(1,I)	REAL	t	s	Independent variable time for ordered pairs RADG
RADG(2,I)	REAL	R	m	Secondary source radius as a function of time
RELHUMID	REAL		%	Ambient relative humidity
RHOA	REAL	$\rho_a$	$\text{kg/m}^3$	Ambient air density
RM	REAL	$R_m$	m	Radius at EMAX (when secondary source mass evolution rate is a maximum)
RMAX	REAL	$R_{\text{max}}$	m	Maximum secondary source radius
RT2	REAL	$\sqrt{2}$	n/a	Constant
R1SS	REAL	$R_p$	m	Steady state primary source radius
R1T(1,I)	REAL	t	s	Independent variable time for ordered pairs R1T

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
R1T(2,I)	REAL	$R_p$	m	Primary source radius as a function of time characterized by ordered pairs
SIGX_COEFF	REAL			Along-wind similarity coefficient
SIGX_MIN_DIST	REAL		m	Minimum distance to apply along-wind dispersion correction
SIGX_POW	REAL		n/a	Along-wind similarity power
SLEN	REAL	L	m	Steady state source length
SQPI02	REAL	$\sqrt{\pi/2}$	n/a	Constant
SQRTPI	REAL	$\sqrt{\pi}$		Constant
SRCDEN(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCDEN
SRCDEN(2,I)	REAL	$\rho$	kg/m <sup>3</sup>	Secondary source density as a function of time
SRCENTH(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCENTH

<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
SRCENTH(2,I)	REAL	h	J/kg	Secondary source enthalpy as a function of time
SRCWA(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWA
SRCWA(2,I)	REAL	w <sub>a</sub>	mass fraction	Secondary source air mass fraction as a function of time
SRCWC(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWC
SRCWC(2,I)	REAL	w <sub>c</sub>	mass fraction	Secondary source contaminant mass fraction as a function of time
SWID	REAL		m	Steady state source width
SZM	REAL	S <sub>z0m</sub>	m	Value of S <sub>z0</sub> at EMAX (when z <sub>0</sub> at secondary source mass evolution rate is a maximum)
TAMB	REAL	T	K	Ambient temperature
TEND	REAL		s	Termination time of secondary source



<u>Variable</u>	<u>Data Type</u>	<u>Symbol</u>	<u>Units</u>	<u>Comments</u>
TINP	CHARACTER*24			Time DEGADISIN was executed
TITLE(1:4)	CHARACTER*80			Text title block 4 lines of 80 spaces
TO(I)	REAL		s	Time of release for observer I
TSURF	REAL	$T_s$	K	Surface temperature
USTAR	REAL	$u_*$	m/s	Friction velocity
UO	REAL	$u_0$	m/s	Ambient velocity at height $z_0$
WTCO	REAL	$F$	$\text{kmol/m}^2\text{satm}$	Mass transfer coefficient when IWTFL=-1
XV(I)	REAL	$x_v$	m	Virtual source position for estimation of $S_y$ in SSG
ZO	REAL	$z_0$	m	Height for velocity $u_0$
ZR	REAL	$z_R$	m	Roughness length

END

10286

DTIC